# On Learning Symmetric Locomotion

Farzad Abdolhosseini
farzadab@cs.ubc.ca
University of British Columbia
Vancouver, Canada

Hung Yu Ling
hyuling@cs.ubc.ca
University of British Columbia
Vancouver, Canada

Zhaoming Xie
zxie47@cs.ubc.ca
University of British Columbia
Vancouver, Canada

Xue Bin Peng
xbpeng@berkeley.edu
University of California
Berkeley, California

Michiel van de Panne
van@cs.ubc.ca
University of British Columbia
Vancouver, Canada

## ABSTRACT

Human and animal gaits are often symmetric in nature, which points to the use of motion symmetry as a potentially useful source of structure that can be exploited for learning. By encouraging symmetric motion, the learning may be faster, converge to more efficient solutions, and be more aesthetically pleasing. We describe, compare, and evaluate four practical methods for encouraging motion symmetry. These are implemented via particular choices of structure for the policy network, data duplication, or via the loss function. We experimentally evaluate the methods in terms of learning performance and achieved symmetry, and provide summary guidelines for the choice of symmetry method. We further describe some practical and conceptual issues that arise. Because similar implementation choices exist for other types of inductive biases, the insights gained may also be relevant to other learning problems with applicable symmetry abstractions.

## CCS CONCEPTS

• **Computing methodologies → Animation**; **Reinforcement learning**.

## KEYWORDS

locomotion, reinforcement learning, symmetry

**ACM Reference Format:**
Farzad Abdolhosseini, Hung Yu Ling, Zhaoming Xie, Xue Bin Peng, and Michiel van de Panne. 2019. On Learning Symmetric Locomotion . In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3359566.3360070


*MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom*

ACM ISBN 978-1-4503-6994-7/19/10…$15.00
https://doi.org/10.1145/3359566.3360070

## 1 INTRODUCTION

Deep reinforcement learning (DRL) has significant potential as a general framework for the control of physically-simulated movement, as achieved via learned control policies that optimize a reward objective. Recent successes of DRL in animation and machine learning show the ability to produce robust learned locomotion for simulated humans, animals, and imaginary legged creatures. However, there remains a compelling need to improve learning efficiency and motion quality for DRL to become a widely-adopted animation tool.

One obvious path towards faster-and-better learning relies on exploiting the motion symmetry that is a common attribute of human and animal locomotion; gait symmetry is an indicator of healthy outcomes in physiotherapy [Riskowski et al. 2011; Robinson et al. 1987]. Asymmetric gaits are often associated with physical injuries and neural impairments such as stroke. A symmetry constraint or symmetry-favoring bias thus offers a readily available and convenient path towards faster learning and more realistic outcomes. It is also largely orthogonal to most other efficiency improvements.

Naively, exploiting symmetry might be expected to yield a 2× learning speedup, and may help to avoid some of the undesired asymmetric local minima that DRL is prone to exploit. On the other hand, it could also be the case that asymmetric policies and motions serve a useful role as an intermediate path towards finding eventual optimal symmetric motions, and therefore hard symmetry constraints may be problematic. Another important subtlety is that while a symmetric policy helps achieve symmetric motions, it does not guarantee a symmetric outcome. For example, a quadruped gallop and a biped lope are asymmetric gait cycles, as each gait cycle begins with a leading left or right foot, while the underlying policy can still be fully symmetric.

What is the best way to integrate a symmetry bias or other forms of symmetry enforcement into the learning process? How much benefit does it offer in terms of learning speed and learning outcomes? What are other considerations for symmetry-informed methods? The principal contribution of our work is an in-depth analysis of four different methods of incorporating symmetry into the learning process:

**DUP** Duplicating tuples with their symmetric counterparts.
**LOSS** Adding a symmetry auxiliary loss.
**PHASE** Motion phase mirroring.
**NET** Enforcing symmetry in the network itself.

Two of these methods are new (DUP, NET) and two are already present in existing literature (LOSS, PHASE), albeit without a systematic evaluation of all the issues around symmetry enforcement. The methods incorporate knowledge of symmetry into the policy structure (NET), the learning data (DUP, PHASE), or via the learning loss (LOSS). We also believe that the results are of more general interest, because they illustrate (and experimentally validate) various ways that inductive biases can be incorporated into DRL methods.

## 2 RELATED WORK

Motion symmetry has been a topic of interest for many years in the study of human motion and movement biomechanics. Symmetric motions are perceived to be more attractive, e.g., for dance [Brown et al. 2005], and gait symmetry is seen as a desirable outcome for physiological manipulation [Robinson et al. 1987]. While symmetry is a common assumption in the study of gait and posture, individual gaits often do exhibit asymmetries due to various possible functional causes [Seeley et al. 2008]. We refer the reader to a past review article [Sadeghi et al. 2000] for insights into the degree of symmetry of lower limbs movement during able-bodied gait and the potential influence of limb dominance on the motion symmetry of the lower extremities and human gaits [Riskowski et al. 2011]. It is also not obvious how to best quantify the asymmetry of human gaits, and thus specific symmetry metrics have been proposed [Hsiao-Wecksler et al. 2010; Viteckova et al. 2018].

The robust control of physics-based character locomotion has been a long-standing challenge for character animation. We refer the reader to a survey paper for a detailed history [Geijtenbeek and Pronost 2012]. An early and enduring approach to controller design has been to structure control policies around finite state machines (FSMs) and feedback rules that use a simplified abstract model or feedback law. These general ideas have been applied to human athletics, running [Hodgins et al. 1995], and a rich variety of walking styles [Coros et al. 2010; Lee et al. 2010; Yin et al. 2007]. Many controllers developed for physics-based animation further use optimization methods to improve controllers developed around an FSM-structure, or use an FSM to define phase-dependent objectives for an inverse dynamics optimization to be solved at each time step. Policy search methods, e.g., stochastic local search or CMA [Hansen 2006], can be used to optimize the parameters of the given control structures to achieve a richer variety of motions, e.g., [Coros et al. 2011; Yin et al. 2007], and efficient muscle-driven locomotion [Wang et al. 2009]. Many of the FSM controllers use hard-coded symmetries, which assign the roles of stance-leg or swing-leg to the left and right legs, as a function of the FSM state. It is common in kinematic-based approaches to also mirror all the available motion data in order to double the effective size of the data set, and to reflect the often-symmetric nature of human locomotion, e.g., [Bruderlin and Calvert 1989; Holden et al. 2017]. Lastly, trajectory optimization-based methods also commonly assume motion symmetry when convenient, e.g., [Majkowska and Faloutsos 2007].

More recently, locomotion synthesis has attracted significant attention from the reinforcement learning (RL) community, where the OpenAI Gym tasks have become a popular RL benchmark [Brockman et al. 2016]. In this context, symmetry constraints are commonly not imposed, and the resulting motions often have noticeable asymmetries. Further work extends these efforts in a variety of ways, including traversing challenging terrains [Heess et al. 2017]. More realistic and dynamic motions can be achieved with the help of motion-capture clips [Peng et al. 2018, 2017] and these use what we refer to as the PHASE symmetry method, with the goal of more efficient learning. [Liu et al. 2016] uses a variation of PHASE in which individual strides (half steps) are mirrored and concatenated to generate symmetric reference motions. However, there exist no robust documented experiments to verify the efficiency gains. The efficient learning of controllers that are capable of producing high-quality motion for realistic-strength characters remains a challenging problem in the absence of motion capture data. Recent work makes progress on this problem using RL with a combination of energy optimization, learning curriculum, and an auxiliary motion symmetry loss [Yu et al. 2018], which we shall refer to as the LOSS method.

A recent result investigates how DRL problems can become prone to learning plateaus because of winner-take-all solution modes [Schaul et al. 2019]. These can easily arise in DRL because the distribution of data for policy learning is directly influenced by the policy itself. Thus in the case of multiple diverging decision paths, one of the modes will quickly dominate. The choice of whether to encourage symmetry, and how to do so, may create an optimization landscape that exhibits similar properties.

## 3 BACKGROUND

In this section, we will briefly introduce the learning problem and the relevant notation. In reinforcement learning (RL), we wish to learn an optimal policy for a Markov Decision Process (MDP). The MDP is defined by a tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$, where $S \in \mathbb{R}^n, A \in \mathbb{R}^m$ are the state space and action space of the problem, the transition function $P : S{\times}S{\times}A \to [0, \infty)$ is a probability density function, with $P(s_{t+1} \mid s_t, a_t)$ being the probability density of visiting $s_{t+1}$ given that at state $s_t$, the system takes action $a_t$. The reward function $r : S{\times}A \to \mathbb{R}$ gives a scalar reward for each transition of the system. $\gamma \in [0, 1]$ is the discount factor. The goal of reinforcement learning is to find a parameterized policy $\pi_\theta$, where $\pi_\theta : S \times A \to [0, \infty)$ is the probability density of $a_t$ given $s_t$, that solves the following optimization problem:

$$\max_\theta J_{RL}(\theta) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].$$

A class of algorithms to solve this problem is policy gradient [Sutton et al. 1999]. Proximal Policy Optimization (PPO) [Schulman et al. 2017] is a popular variant of policy gradient for training locomotion tasks. At each PPO learning iteration, the agent interacts with the environment and collects a set of $(s_t, a_t, r_t, s_{t+1})$ tuples. These are then used to estimate the gradient of the PPO surrogate loss, $L_{PPO}$, and standard stochastic descent algorithms can be used to update the network parameters $\theta$ to minimize this surrogate loss.

# 4 SYMMETRY ENFORCEMENT METHODS

We now describe four methods for enforcing symmetry, using duplicate tuples, auxiliary losses, a time-indexed motion phase, and architecture-based methods. We begin by formally defining symmetric trajectories and symmetric policies. Two trajectories are symmetric if for each state-action tuple, $(s, a)$, from one trajectory, the corresponding state-action tuple is given by $(\mathcal{M}_s(s), \mathcal{M}_a(a))$ for the other trajectory, where $\mathcal{M}_s$ and $\mathcal{M}_a$ are defined as follows,

$$\begin{array}{ll} M_s : \mathcal{S} \to \mathcal{S} & M_a : \mathcal{A} \to \mathcal{A} \\ M_s(s) = \text{the mirror of state } s & M_a(a) = \text{the mirror of action } a \end{array}$$

Note that the mirroring functions are attributes of the environment and not attributes of the enforcement method or learning pipeline. Here we use *environment* to refer to the combination of the character, its simulated world, and the task, as is common in RL settings. All the symmetry enforcement methods we shall describe require both of these functions as a minimum requirement. Similarly, we can define a symmetric policy to be one where the following holds for all states $s \in \mathcal{S}$:

$$\pi_\theta(M_s(s)) = M_a(\pi_\theta(s)). \tag{1}$$

A symmetric policy thus produces the mirrored action when given the mirrored state as input. RL methods such as PPO also use state-value functions during the learning process. The output of these value functions should remain unchanged for any state and its mirrored counterpart. The construction of the mirror functions for our environments (Section 6) is further elaborated in Appendix A.1.

The methods discussed in this section attempt to achieve symmetric gaits by encouraging or constraining the learned policies to be symmetric. However, even if successful, this may be insufficient to guarantee a symmetric gait. In particular, a symmetric policy may learn to favour motions with staggered poses, where the dominant foot is always in front. This may confer advantages with respect to balance and agility. Once such a policy is initialized to an initial asymmetric staggered pose, it can continue with an asymmetric motion. With regard to the policy, it is not always possible to achieve exact symmetry in a parameterized model such as a neural network. For example, regions of the state space may remain unexplored during the learning process, and thus symmetry cannot be enforced for such regions. Therefore, the equality in Equation 1 is not always assumed to be strict.

It is possible to directly optimize for gait symmetry with reinforcement learning by including quantitative symmetry measures in the reward function, such as the Symmetry Index [Robinson et al. 1987] or other measures [Viteckova et al. 2018]. However, we share the sentiment of previous work [Yu et al. 2018] that directly optimizing such measures may be ineffective, as they introduce delayed or sparse rewards that may make the learning problem more difficult. Consequently, our work focuses on methods that can be used for obtaining approximately symmetric policies, which are described in the remainder of this section.

## 4.1 Duplicate Tuples (DUP)

This method may be the most intuitive way for achieving symmetry and is a form of data augmentation. In this approach each trajectory tuple is duplicated, mirrored, then added as a valid experience tuple along with the original. More formally, let $\tau = (s_1, a_1, r_1, \ldots, s_T)$ be a trajectory sampled from the environment. A post-processing step will compute the mirrored trajectory of $\tau$, i.e. $\tau' = (M_s(s_1), M_a(a_1), r_1, \ldots, M_s(s_T))$, and both $\tau$ and $\tau'$ will be added to the roll-out memory buffer for learning. Notice that the rewards, $r_1, \ldots, r_{T-1}$ are the same in both $\tau$ and $\tau'$. This is because the reward function $r(s, a)$ is automorphic under the symmetry transformation, namely $r(s, a) = r(\mathcal{M}_s(s), \mathcal{M}_a(a))$.

One drawback of using this approach is that the mirrored tuples are not strictly on-policy, as assumed by policy-gradient RL methods. Thus it could be problematic when used with methods such as PPO [Schulman et al. 2017] and TRPO [Schulman et al. 2015]. The off-policy issue arises because at training time the policy $\pi_\theta$ is not guaranteed to be symmetric, and therefore the probability of sampling action $M_a(a_t)$ from $\pi_\theta(M_s(s_t))$ could be low, effectively corresponding to an off-policy action. However, our results show that this is not necessarily a critical issue in practice.

## 4.2 Auxiliary Loss (LOSS)

In this method proposed by Yu et al.[Yu et al. 2018], the authors create a symmetry loss defined as follows:

$$L_{sym}(\theta) = \sum_{t=1}^{T} \|\pi_\theta(s_t) - M_a(\pi_\theta(M_s(s_t)))\|^2 \tag{2}$$

and optimize this as an auxiliary loss in addition to the default PPO loss:

$$\pi_\theta = \operatorname*{argmin}_\theta L_{PPO}(\theta) + wL_{sym}(\theta), \tag{3}$$

where $w$ is a scalar hyper-parameter used to balance the gait symmetry loss with the standard policy optimization loss which aims to maximize the original objective. The authors use $w = 4$ for their results. An alternative approach would be to simply include the symmetry loss as an extra reward term. However, the auxiliary loss is generally preferable; the loss term is differentiable and therefore provides a clear signal to optimize rather than being included via the PPO-approximated gradient. Changing the reward function may also induce unexpected behaviours.

Yu et al. [2018] showed improvements in the sample efficiency for their four tasks with a factor of approximately two (see Figure 8 in [Yu et al. 2018]). However, the symmetric loss is shown to be beneficial only in the context of a given curriculum learning algorithm; in its absence, there was no significant improvement over a vanilla-PPO baseline, and in one case (the humanoid) using the symmetric loss proved to be detrimental (please refer to the same plot). The addition of an extra hyper-parameter may generally be seen as undesirable. However, in practice, we find in our experiments that the method is not very sensitive to the choice of $w$ and we end up using the default value in all settings.

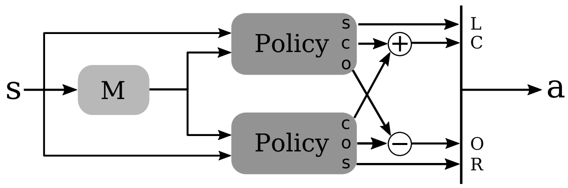


**Figure 1: An universal method for converting any neural network into a symmetric network policy. $\mathcal{M}$ block is an environment-dependent state mirroring function. The two policy blocks are the same neural network module, with the output terminals re-ordered for illustration clarity. The *s, c, o* terminals corresponds to *side, common,* and *opposite* joints as described in Appendix A.1.**

## 4.3 Phase-Based Mirroring (PHASE)

To study locomotion, the gait can usually be divided into repeated *gait cycles*, which can then further be parameterized using a phase variable $\phi \in [0, 1)$, which then wraps back to $\phi = 0$ upon reaching $\phi = 1$. A common assumption is to advance the phase linearly with time. Another strategy that can help provide additional robustness is to perform a phase-reset at each bipedal foot strike, e.g., set $\phi = 0$ upon left-foot strike and $\phi = 0.5$ upon right foot strike. To enforce symmetry, a policy is only learned for the first half cycle, and is replaced by the policy with mirrored states-and-actions during the second half cycle:

$$a_t = \begin{cases} \pi_\theta(s_t) & 0 \le \phi(s_t) < 0.5 \\ M_a(\pi_\theta(M_s(s_t))) & 0.5 \le \phi(s_t) < 1 \end{cases} \tag{4}$$

In our experiments, we strictly advance the motion phase as a function of time and we do not implement phase-resets. For forward-progress tasks, this then corresponds to providing a mandated duration for each half-cycle of the motion. The phase-based method is particularly useful for imitation-guided learning scenarios such as those presented in [Peng et al. 2017], [Peng et al. 2018], and [Xie et al. 2019]. The goal in these cases is to imitate a reference motion capture clip with the help of a phase-indexed reward that measures the distance from the reference motion. The use of the PHASE symmetry in that context is motivated by the potential for faster learning.

The PHASE approach is simple to implement and does not require modifying training in any way since it can be implemented directly within the environment. However, the potential for abrupt changes exists at $\phi = 0.5$ when the phase is strictly computed as a function of time.

## 4.4 Symmetric Network Architecture (NET)

Another approach towards enforcing symmetry is to impose symmetry at the network architecture level. The goal here is to choose a network architecture such that Equation 1 holds for all states $s$ and all network parameters $\theta$. There are multiple ways to go about designing such an architecture. However, they may require some knowledge about how the actions and/or states in which case having access to the mirroring functions $M_s$ and $M_a$ is strictly-speaking not enough.

A general case description of this method would be lengthy, and thus we focus only on the key aspects here. The simplest case occurs when we can assume that the action vector is simply divided into two, one corresponding to each side of the body, and that the actions of one side can readily be applied to the other side through a simple swapping operation. This ignores the common parts such as the torso and the head for the time being. More concretely, consider:

$$a = \begin{bmatrix} a_l \\ a_r \end{bmatrix}$$
$$M_a(a) = \begin{bmatrix} a_r \\ a_l \end{bmatrix}$$

where $a_l$ and $a_r$ are vectors of equal size. In this case, we can define a symmetric policy composed of an inner network $f$ as follows:

$$\pi_{side}(s) = \begin{bmatrix} f(s, M_s(s)) \\ f(M_s(s), s) \end{bmatrix}$$

It is easy to show in this case that Equation 1 holds:

$$\begin{aligned} \pi_{side}(M_s(s)) &= \begin{bmatrix} f(M_s(s), M_s(M_s(s))) \\ f(M_s(M_s(s)), M_s(s)) \end{bmatrix} \\ &= \begin{bmatrix} f(M_s(s), s) \\ f(s, M_s(s)) \end{bmatrix} \\ &= M_a\left(\begin{bmatrix} f(s, M_s(s)) \\ f(M_s(s), s) \end{bmatrix}\right) \\ &= M_a\left(\pi_{side}(s)\right) \end{aligned}$$

When the action space also includes actions for common parts, i.e., those such as the torso and head that have no symmetric counterparts, it is easy to define $\pi_{com}(s) = h(s) + h(M_s(s))$ which is then invariant to left/right mirroring. Finally, the policy is then a combination of the common actions and the side actions:

$$\pi_\theta(s) = \begin{bmatrix} \pi_{com}(s) \\ \pi_{side}(s) \end{bmatrix}$$

Please refer to Figure 1 for an illustration of the NET method.

A drawback of this method is that it requires knowledge about the state and action symmetry structures to redefine the network. Also, this method is highly sensitive to state and action normalization. The problem is that an ordinary normalization based on past experiences may break the symmetry. Though the other methods introduced here can also suffer from the same problem, this method is much more sensitive to the issue.

## 4.5 Practical Considerations

There are a number of practical considerations to take into account when working with each of the methods introduced in the previous section. In terms of implementation, the DUP and PHASE methods are the easiest to implement as they required little to no change to the learning pipeline. Architecture-based mirroring (NET) requires the most modification to both the learning pipeline and the environments. The LOSS method is the only approach here that allows us to balance the desire for symmetry with the original learning objective, albeit at the cost of an extra hyper-parameter. The NET method produces a truly symmetric policy, which is not possible with the other methods. The PHASE method is the approach best suited for coping with neutral states, which represent symmetric states where it may become problematic to break symmetry. We revisit this point later. PHASE is also restrictive in that it enforces a predefined walk cycle timing.

One more consideration relates to the application of normalization to network inputs, which is commonly done by using statistics gathered from the data itself. However, this can break some of the mirroring assumptions. The problem is most severe when using a symmetric network architecture, although other methods are also impacted. Fortunately, developing a normalization scheme that works correctly is relatively straightforward. A simple approach is to duplicate the states (or actions) as in Section 4.1 and to compute the statistics based on the aggregated set of states (or actions) and their mirrored states (or actions).

## 5 GAIT SYMMETRY METRICS

All of the methods discussed only provide indirect paths, via the learned policies, for achieving symmetry for the actual motions. Therefore it is important to evaluate how well these methods do at achieving their final goal. Yu et al.[Yu et al. 2018] use an established metric in the biomechanics literature known as the Robinson Symmetry Index (SI):

$$SI = \frac{2|X_R - X_L|}{X_L + X_R} \cdot 100, \tag{5}$$

where $X_R$ is a scalar features of interest, such as the duration of the stance phase for the right leg, and $X_L$ is its counterpart for the left leg. Previous work using the LOSS method [Yu et al. 2018] chooses to use the average actuation magnitude as the parameter of interest which leads to $X_R = \sum_{t=1}^{T} \|\boldsymbol{\tau}_{t,R}\|_2$ where $\boldsymbol{\tau}_{t,R}$ is the vector of applied torques at time $t$ for the right leg. We will refer to this as the *actuation symmetry index (ASI)*. In practice, we found that the ASI can be misleading in some circumstances, e.g., a high torque applied to the right hip can be conflated with a high torque applied to the left knee, which is not desirable. ASI also loses information about signs of the applied torques.

The *phase-portrait* is another tool that can be used to qualitatively investigate the symmetry or asymmetry of a gait, as seen in [Hsiao-Wecksler et al. 2010]. The phase-portrait is a scatter plot drawn over a period of time, usually over a single gait cycle. The $x$ and $y$-axes of the 2D plot correspond to the position and velocity, respectively, of a joint of interest, such as the hip flexion, For an asymmetric gait, the phase portraits of the two sides will not fully overlap. To numerically quantify the similarity between two phase-portraits, we propose to use a *phase-portrait index* (PPI). One problem to address is that the left and right limbs usually have a phase offset even for a symmetric motion. This is not a problem when inspecting the phase-portraits visually, but the problem needs to be addressed to compute a meaningful metric. We solve this by finding the best phase offset between the left and the right side through an exhaustive search. We also normalize each axis so that $x, y \in [-1, 1]$ to address the potential discrepancy between magnitudes of different gaits. The final PPI is defined according to:

$$PPI = \frac{1}{C} \min_{s} \sum_{t=0}^{C-1} \|q_t^R - q_{t+s}^L\|_1 + \|\dot{q}_t^R - \dot{q}_{t+s}^L\|_1, \tag{6}$$

where $C$ is the length of a gait cycle, $q_t^R$ and $\dot{q}_t^R$ are the normalized right joint position and velocity at time $t$. Similarly, $q_{t+s}^L$ is the normalized left joint position at time $t+s$ modulo $C$, as the elements that are shifted beyond the last position are reintroduced at the beginning.

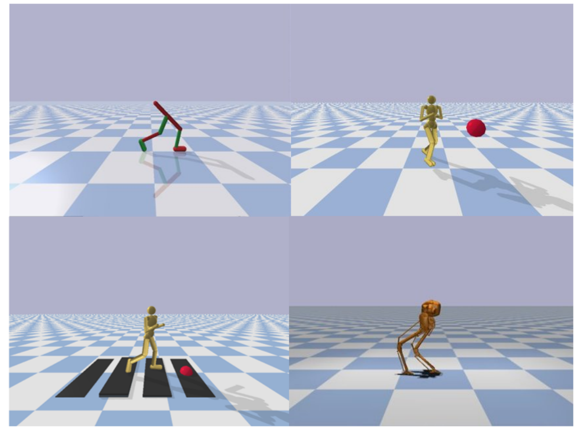

**Figure 2: Top-left:** *Walker2D*. **Top-right:** *Walker3D*. **Bottom-left:** *Stepper*. **Bottom-right:** *Cassie*.

## 6 ENVIRONMENTS

We evaluate the effectiveness of the enforcement methods described in Section 4 on four different locomotion tasks, i.e., RL “environments”. The environments were chosen to represent a fairly diverse range of locomotion tasks. They are described in detail below. For each environment, we run each method 5 times and plot the mean results.

*Walker2D.* The implementation of *Walker2D* environment is taken directly from PyBullet [Coumans and Bai 2019] without further modification. The purpose of this environment is to evaluate each symmetry method on a well-established existing reinforcement learning environment. The task is for the character to walk as far as possible in the forward direction in the allotted time. An action is a 6D vector corresponding to a normalized torque at each of the hip, knee, and ankle on both left and right legs. The observation space is 22D and consists of root information (root z-coordinate, x and y heading vector, root velocity, roll, and pitch), joint angles, joint angular velocities, and binary foot contact information.

*Walker3D.* This represents a 3D character simulated in PyBullet, with targets randomly placed, at a distance, in the half-plane in front of the character. The task requires character to navigate towards the target and then stop at the target. A new target will be chosen, in the forward half-plane of the current character orientation, once the target is reached and one second has passed. The 3D character has 21-DoF corresponding to abdomen (x3), hip (x3), knee, ankle, shoulder (x3), and elbow. The observation space is 52D, and is analogous to that provided for *Walker2D*, with an additional 2D vector representing the target location in the character root frame.

*Stepper.* *Stepper* uses the same model as *Walker3D*, and requires it to navigate terrain consisting of a sequence of stepping blocks. The blocks are randomly generated by sampling from the following distributions: spacing $d \sim \mathcal{U}(0.65, 0.85)$ meters and height variation of the next step $h \sim \mathcal{U}(\text{-}25, 25)^{\circ}$. The character receives information for two upcoming blocks as an $(x, y, z)$ offset in character root space. The stepping block information advances when either foot contacts the immediate next block, which effectively forces the character to step precisely on each step. The precise foot placement requirement, as well as variable terrain height, makes this environment more challenging than *Walker3D*.

*Cassie.* The task requires a bipedal robot Cassie to walk forward at a desired speed while mimicking a reference motion. Since the reference motion is time-indexed, the character receives a phase variable as input. The phase variable varies according to $\phi \in [0, 1)$ in the gait cycle. In addition to phase, the character receives other inputs including the height, the orientation expressed as a unit quaternion, pelvis velocities, angular velocities, and acceleration, joint angles and angular velocities. In total, the Cassie robot has a 10D action space and 47D observation space. Another important distinction between *Cassie* and the other tasks is that it is implemented in MuJoCo [Todorov et al. 2012], while other environments use the Pybullet [Coumans and Bai 2019] physics engine. This simulated model has also been validated to be close to the physical Cassie robot [Xie et al. 2019].

## 7 RESULTS

We compare the four methods, together with an asymmetric baseline, across four different locomotion tasks of varying difficulties. The source code is available at https://github.com/UBCMOCCA/SymmetricRL.

### 7.1 Summary

We begin with a high-level summary of our findings. All symmetry enforcement methods improve motion quality over the baseline, but they cannot be reliably ranked across different environments. In general, DUP is the least effective in enforcing symmetry, while LOSS is the most consistent. For imitation-guided tasks, where the reward is related to imitating a time-indexed reference motion, such as for *Cassie* and *DeepMimic*, the PHASE method appears to be superior.

Regarding learning speed, the symmetry enforcement methods have no consistent and predictable impact, positive or negative. While this contradicts our initial expectation, it does not provide the full picture. In particular, even though BASE achieves relatively high rewards in *Stepper*, it was unable to make forward progress in any of the five runs. In summary, we suggest symmetry methods be used for producing higher quality symmetric motions, i.e., closer to what we might expect from human and animal movement, but not necessarily for faster learning. We further expand the comparison of the different methods in two sections below.

### 7.2 Effect on Learning Speed

One of our initial hypotheses was that the learning speed can be improved by enforcing symmetry. Symmetry can be considered as domain knowledge that may otherwise be difficult to learn, especially considering its abstract nature. However, our experiments indicated that enforcing symmetry has no consistent impact on the learning speed. As shown in Figure 3, BASE performs well in *Walker2D* and *Walker3D*. In particular, although BASE was not initially the fastest in *Walker3D*, it ultimately achieves a higher return than all mirroring methods. On the other hand, BASE fails to learn the *Stepper* task in all five runs; it often pauses near the beginning without taking a single step. This is consistent with findings by Yu et al., who also find that symmetry enforcement can be crucial when learning more difficult tasks.

For *Cassie*, the benefit of enforcing symmetry is evident because the reward explicitly encourages the character to imitate a symmetric reference motion. We hypothesize that for such a case, symmetry constrains the search space in a suitable way for the symmetric task. However, if symmetry is not rewarded, explicitly or implicitly, then its effects may not be reflected in the learning curve. Finally, between the symmetry methods, there is no clear winner in terms of learning speed.

*PHASE and Imitation-Guided Learning.* In phase-based symmetry experiments, we define a phase variable in correspondence to the gait cycle. For the *Cassie* environment, we use a period of 0.8 s, which is determined based on the reference motion. For all other environments, we assign a period based on a working solution.

We find phase-based symmetry enforcement to be effective for imitation-guided learning, as it outperforms other methods by a significant margin for *Cassie*. When comparing *Cassie* with *DeepMimic* [Peng et al. 2018], which also uses an imitation objective, we find the results to be consistent. The learning curves for our *DeepMimic* symmetry experiment are presented in Appendix A.3. We hypothesize that phase-based symmetry is effective for imitation-guided tasks when the motion clips used for training containing suitably-periodic and symmetric motions. On the other hand, PHASE constrains the period of the gait cycle, which can be harmful for non-imitation tasks. PHASE performs poorly in terms of learning speed when used without a reference motion, i.e., for *Walker2D*, *Walker3D*, and *Stepper*, although it can do well in terms of quality, e.g., for *Walker3D*.

*Alternate Symmetric Network.* The NET method presented in Figure 1 is an intuitive way of converting any neural network into a symmetric policy. However, it is perhaps not the immediate solution that one would come up with when tasked to design a symmetric neural network. We include one of our earlier constructions of symmetric policy in Appendix A.2, which we refer to as NET-ALT. A major difference between NET and NET-ALT is that the latter uses shared weights at the layer level to explicitly enforce the symmetry constraint in Equation 1. Despite this, the two architecture-based mirroring methods should, in theory, have similar performance. As can be seen in Figure 3, NET-ALT significantly outperforms NET in the *Walker2D* environment, along with the baseline and all other mirroring methods. We believe that the structure of the symmetric layer matrix in Appendix A.2 may be the key to resolve this gap, which remains to be verified.

*Policy Network Ablation Study.* As an ablation study, we removed the symmetry constraint for the value network in the NET method. Since our goal is to produce a symmetric policy, and the value network is discarded after training, we want to see how enforcing symmetry in the value network during training affects the learning speed. In Figure 3, the two curves of interest are NET and NET-POL,

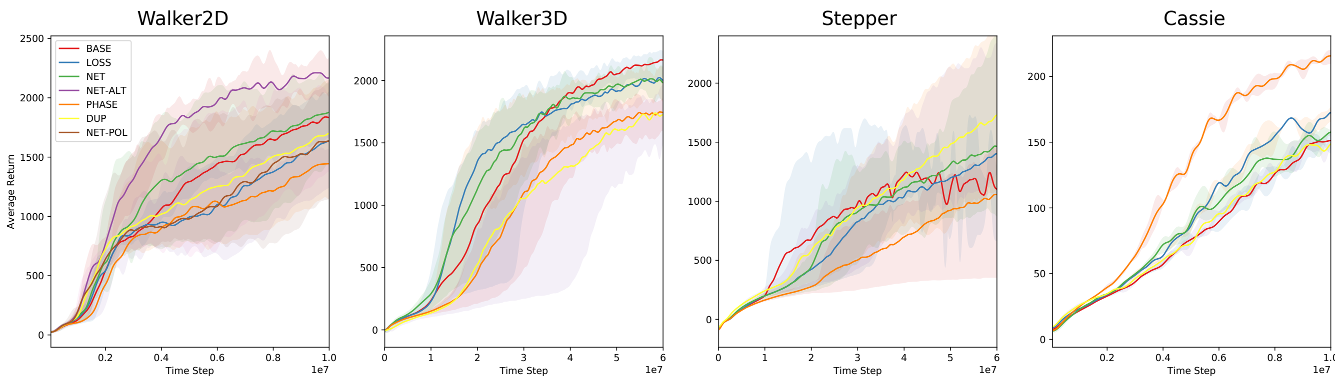


**Figure 3: Learning curves for different symmetry methods in each of the four locomotion environments (Section 6). The *Walker2D* plot contains two additional experiments aside from the baseline and four symmetry methods. *NET-ALT* uses an alternate formulation of symmetric network architecture described in Appendix A.2. *NET-POL* is an ablation study on *NET* with symmetry enforcement only on the policy network and not on the value network.**

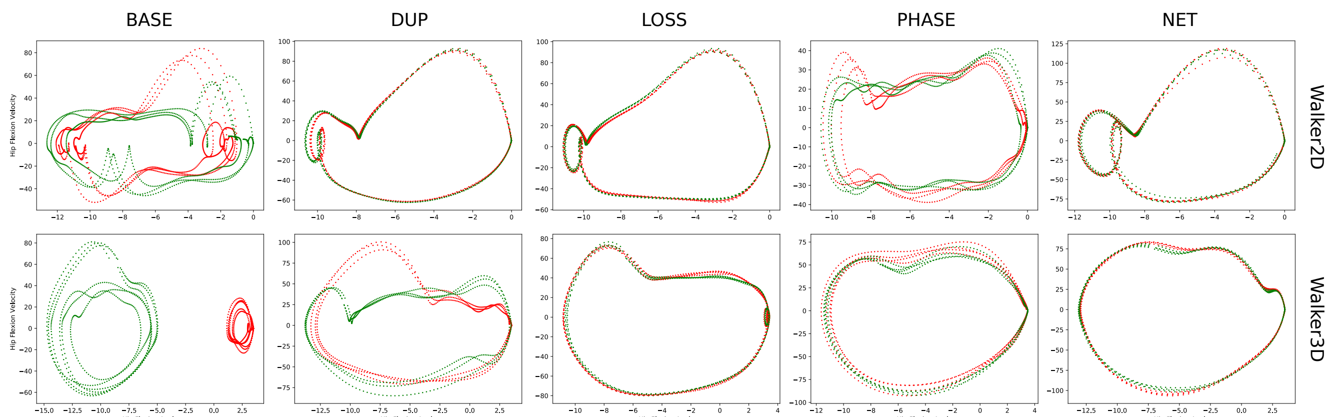


**Figure 4: Phase-portrait for *Walker2D* and *Walker3D*. The green curve is for the left hip flexion and red for the right side. The more symmetric the motion, the more aligned are the curves.**

where the latter has the symmetry constraint removed for the value network. Our experiment shows that it is beneficial to enforce the symmetry constraint for the value network during training since the difference between the curves is not insignificant.

## 7.3 Symmetry Enforcement Effectiveness

Although learning speed is a major point of interest from the ML perspective, our work is nevertheless motivated by the aesthetics of symmetric gaits that are needed for applications in animation. We measure the effectiveness of each symmetry enforcement method on the metrics we defined in Section 5. In most cases, we find that symmetric gaits are better achieved when any of the enforcement methods are applied, as compared to the baseline. The motions produced by the symmetry methods are also more natural-looking, subjectively speaking, than without mirroring (see supplementary video).

Figure 4 shows the phase-portraits for *Walker2D* and *Walker3D*. The symmetry metrics for all environments are summarized in Table 1 and Table 2. To perform consistent measurement for the metrics, we omit the first two strides in order to limit the influence of the transition period from standing to locomotion. The reported metrics are calculated from the median of the ten subsequent strides after the initial two. For the *Stepper* tasks, we use the median from five strides to accommodate for the increased difficulty. Also, note the *Stepper* results are missing for BASE because it was unable to produce consistent gait cycles that can be measured. In most cases, the policy either learns to pause at the starting location or falls after taking one or two steps.

As in learning speed, there is not a single best mirroring method across all environments. However, from the overall picture, we found that LOSS and PHASE to be the most consistent among all methods. In general ASI and PPI do not agree on a single best method except for the *Cassie* task where PHASE is the best.

**Table 1: Actuation SI. Lower numbers are better.**

| | Walker2D | Walker3D | Stepper | Cassie |
|---|---|---|---|---|
| BASE | *3.97* | 6.36 | ✗ | 9.27 |
| DUP | 3.77 | 7.57 | 7.54 | 6.58 |
| LOSS | 2.56 | 4.48 | 6.36 | *15.72* |
| PHASE | 3.77 | **2.55** | **3.99** | **4.49** |
| NET | 2.00 | *10.64* | *28.97* | 5.15 |
| NET-ALT | **1.04** | – | – | – |
| NET-POL | 1.71 | – | – | – |

**Table 2: Phase-portrait index. Lower numbers are better.**

| | Walker2D | Walker3D | Stepper | Cassie |
|---|---|---|---|---|
| BASE | *1.06* | *2.16* | ✗ | *0.49* |
| DUP | 0.39 | 1.61 | 0.57 | 0.41 |
| LOSS | 0.33 | **0.19** | **0.46** | 0.31 |
| PHASE | 0.57 | 0.30 | 0.49 | **0.17** |
| NET | **0.16** | 0.58 | *0.65* | 0.23 |
| NET-ALT | **0.16** | – | – | – |
| NET-POL | 0.28 | – | – | – |

## 8 DISCUSSION

Symmetry can sometimes be harmful, especially when the character begins from or otherwise arrives at a neutral pose, i.e., a symmetric pose where $s = M_s(s)$. The problem is that a symmetric policy is incapable of escaping from a neutral pose since the action that it takes would also be symmetric. When a symmetric action is applied in a symmetric state, the next state is necessarily also symmetric. For instance, a character that starts from the T-pose will likely perform some kind of hopping gait, since the feasible locomotion possibilities which perpetuate symmetric states and actions are limited. To make matters worse, states near the neutral states can also become problematic.

The breaking symmetry problem is most severe when enforcing symmetry through network architecture, as this method is guaranteed to produce true symmetric policies. While DUP and LOSS methods can suffer from the same issue, they can implement workarounds at an additional cost. This issue, however, does not affect PHASE. A simple workaround to this problem is to always start the character from a non-neutral position. This can be easily achieved by adding some random noise to each joint of the initial pose at the start of the task. In practice, we did notice that on occasion the character would converge on a hopping gait. However, the simple workaround works well for the majority of cases in our experiments.

Our work is motivated by the premise that healthy human gaits are usually symmetric. However, this still remains a controversial issue in the biomedical literature [Riskowski et al. 2011; Sadeghi et al. 2000]. The strongest argument for asymmetry in human motor control is the general belief that humans have a dominant side that is often the preferred choice for manipulating objects. This is also tied with the need for a leading foot to start a walk or run cycle in the neutral state problem. One should, therefore, be aware of the implications when enforcing perfect symmetry. Quadrupedal locomotion, which has six commonly observed gaits as opposed to the three gaits of bipeds [McMahon 1984], is also interesting to examine. Of these six, half are fairly symmetric including walk, trot, and rack. However the remaining three, also known as the in-phase gaits which are used at high speeds, are often asymmetric. Since the symmetry of gait and policy are not the same, it would be interesting to see whether it is possible to nevertheless achieve these non-symmetric quadrupedal gaits with a symmetric policy.

## 9 CONCLUSIONS

In this paper, we explore the use of symmetry constraints for DRL-based learning of locomotion skills. We compared four different enforcement methods, in addition to a symmetry-free baseline, across four different locomotion tasks of varying difficulty. We find that enforcing symmetry constraints can in fact sometimes be harmful to learning efficiency, but that in general it produces higher quality motions. When comparing the symmetry methods, we find that the results, both in terms of learning speed and motion symmetry, to be environment-dependent. A notable exception is that the phase-based mirroring method generally performs better than the the baseline for imitation-guided reward settings such as for *Cassie* and *DeepMimic*.

The difference between the enforcement methods is more pronounced from the implementation standpoint. LOSS and PHASE methods have the burden of an additional hyperparameter to tune. However, the additional parameter can also be viewed as an advantage in terms of flexibility. In LOSS, the hyperparameter can be used to adjust the strength of symmetry constraint. For PHASE, the phase variable allows us to define a desired locomotion period. Given the similarities across all methods, it is perhaps justifiable to choose one based on the implementation overhead. DUP is the easiest to implement and evaluate since it requires minimal change to existing RL pipeline and has no hyperparameter to tune. Finally, if the application requires absolute symmetry, then the NET method is guaranteed to produce a symmetric policy.

The application of symmetric policies is not limited to locomotion. Many classical control tasks may benefit significantly from leveraging symmetry, including acrobat, cart-pole, and pendulum [Brockman et al. 2016]. Furthermore, the notion of symmetry extends beyond left-right symmetry and even character motion. The Sudoku game is an example task that exhibits multiple types of symmetry properties. Whether a learning method can take full advantage of all the symmetries remains an open question. However, this paper lays a foundation for enabling future studies on inductive biases based on symmetry.

## REFERENCES

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540
William M Brown, Lee Cronk, Keith Grochow, Amy Jacobson, C Karen Liu, Zoran Popović, and Robert Trivers. 2005. Dance reveals symmetry especially in young men. *Nature* 438, 7071 (2005), 1148.
Armin Bruderlin and Thomas W Calvert. 1989. Goal-directed, dynamic animation of human walking. In *ACM SIGGRAPH Computer Graphics*, Vol. 23. ACM, 233–242.
Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized Biped Walking Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 130.
Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. 2011. Locomotion Skills for Simulated Quadrupeds. *ACM Transactions on Graphics* 30, 4 (2011), Article 59.
Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Thomas Geijtenbeek and Nicolas Pronost. 2012. Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2492–2515.

Nikolaus Hansen. 2006. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation*. 75–102.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. 2017. Emergence of Locomotion Behaviours in Rich Environments. *CoRR* abs/1707.02286 (2017). arXiv:1707.02286 http://arxiv.org/abs/1707.02286

J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of SIGGRAPH 1995*. 71–78.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Elizabeth T. Hsiao-Wecksler, John D. Polk, Karl S. Rosengren, Jacob J. Sosnoff, and Sungjin Hong. 2010. A Review of New Analytic Techniques for Quantifying Symmetry in Locomotion. *Symmetry* 2 (2010), 1135–1155.

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-Driven Biped Control. *ACM Transctions on Graphics* 29, 4 (2010), Article 129.

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (2016).

Anna Majkowska and Petros Faloutsos. 2007. Flipping with physics: motion editing for acrobatics. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 35–44.

Thomas A. McMahon. 1984. *Muscles, Reflexes, and Locomotion* (1 ed.). Princeton University Press, Princeton, New Jersey.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018)* 37, 4 (2018).

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)* 36, 4 (2017).

Jody L Riskowski, Thomas J Hagedorn, Alyssa B Dufour, Virginia A Casey, and Marian T Hannan. 2011. Evaluating gait symmetry and leg dominance during walking in healthy older adults. *Institute of Aging Research, Hebrew SeniorLife, Boston, Usa, Harvad Medical School, USA, School of Public Health, Boston University, Boston* (2011).

RO Robinson, W Herzog, and BM Nigg. 1987. Use of force platform variables to quantify the effects of chiropractic manipulation on gait symmetry. *Journal of manipulative and physiological therapeutics* 10, 4 (1987), 172–176.

Heydar Sadeghi, Paul Allard, François Prince, and Hubert Labelle. 2000. Symmetry and limb dominance in able-bodied gait: a review. *Gait & Posture* 12, 1 (2000), 34–45. https://doi.org/10.1016/S0966-6362(00)00070-9

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. 2019. Ray Interference: a Source of Plateaus in Deep Reinforcement Learning. *CoRR* abs/1904.11455 (2019). arXiv:1904.11455 http://arxiv.org/abs/1904.11455

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *CoRR* abs/1502.05477 (2015). arXiv:1502.05477 http://arxiv.org/abs/1502.05477

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

Matthew K Seeley, Brian R Umberger, and Robert Shapiro. 2008. A test of the functional asymmetry hypothesis in walking. *Gait & posture* 28, 1 (2008), 24–28.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)*. MIT Press, Cambridge, MA, USA, 1057–1063. http://dl.acm.org/citation.cfm?id=3009657.3009806

E. Todorov, T. Erez, and Y. Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

Slavka Viteckova, Patrik Kutilek, Zdenek Svoboda, Radim Krupicka, Jan Kauler, and Zoltan Szabo. 2018. Gait symmetry measures: A review of current and prospective methods. *Biomedical Signal Processing and Control* 42 (2018), 89–100. https://doi.org/10.1016/j.bspc.2018.01.013

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2009. Optimizing Walking Controllers. *ACM Transctions on Graphics* 28, 5 (2009), Article 168.

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan W. Hurst, and Michiel van de Panne. 2019. Iterative Reinforcement Learning Based Design of Dynamic Locomotion Skills for Cassie. *CoRR* abs/1903.09537 (2019). arXiv:1903.09537 http://arxiv.org/abs/1903.09537

KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transctions on Graphics* 26, 3 (2007), Article 105.

Wenhao Yu, Greg Turk, and C. Karen Liu. 2018. Learning Symmetric and Low-energy Locomotion. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018 - to appear)* 37, 4 (2018).

# A SUPPLEMENTARY MATERIAL

## A.1 Mirroring Functions

The mirroring functions, $\mathcal{M}_s$ and $\mathcal{M}_a$ as described in Section 4 are properties of the environment. Consequently, the environment is responsible for providing the necessary information for policies to perform the mirroring operation on state and action. Although the mirroring functions can be arbitrarily complex, we found that all the environments in Section 6 share a similar construction. Using *Walker3D* as an example, the method for deriving mirror functions are described in detail below.

The *Walker3D* character has a total of 21-DoF and each DoF is modelled as a one-dimensional hinge joint. Furthermore, let the $x$-axis be the forward direction and the $z$-axis pointing up in the local coordinate frame of the character. For mirroring purposes, the joints can be divided into three categories, common, opposite, and side. The common categories contain joints that are unchanged by the mirroring function, such as $abdomen_y$. In general, joints that rotate about the $y$-axis should remain unchanged after mirroring. The opposite categories contain joints that are mainly on the torso of the character and they need to be negated for mirroring. In the case of *Walker3D*, only $abdomen_x$ and $abdomen_z$ would fall under this category. The side categories contain joints that are on the limbs. Importantly, for each joint on one side, there must be a corresponding joint on the other side; for instance, the right knee corresponds to the left knee. With the one-to-one mapping, $\mathcal{M}_a$ can simply interchange the applied torques for the respective joints on either side. We found that it is more straightforward if the joint rotation axes are flipped, except for axes aligned on the $y$-axis, for the left and right limbs. Otherwise, additional negation operations need to be applied after interchanging left and right actions.

$\mathcal{M}_s$ follows a similar pattern as described above. For state information that is derived from the character, such as joint angles and angular velocities, the mirrored counterpart would have negated and interchanged values. In addition, the environment may provide additional information, such as character orientation, velocity, and target location in character root space, as in *Walker3D*. For vector-valued information, such as velocity and target location, the values along the $y$-axis should be negated; for orientations, values representing roll and yaw should be negated.

## A.2 Alternate Symmetric Network Architecture

In Figure 1, we presented a universal method for embedding any neural network into a symmetric policy. The NET method effectively uses the same policy module twice with flipped inputs for $s$ and $M(s)$. While this construction is relatively simple to implement, alternative symmetric policy constructions do exist. In this section, we describe the construction used for NET-ALT in Figure 3.

Recall that a symmetric policy is one that satisfies Equation 1, along with the fact that our mirror functions (subsection A.1) essentially perform negation and swapping operation on the state and action vectors. Let us then consider the individual layers of a neural network as matrix operations, in particular, before the application of non-linear activation functions. The full matrix form of the first layer for $s$ and $\mathcal{M}_s(s)$ can be written as,

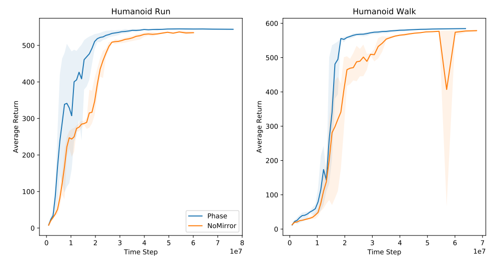


**Figure 5: Learning curves for the original *DeepMimic* environment. BASE and *Phase* corresponds to the symmetry enforcement methods in Figure 3.**

$$\begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix} = (a_{ij})_{4\times4} \begin{bmatrix} C \\ O \\ R \\ L \end{bmatrix} \text{ and, } \begin{bmatrix} W \\ -X \\ Z \\ Y \end{bmatrix} = (a_{ij})_{4\times4} \begin{bmatrix} C \\ -O \\ L \\ R \end{bmatrix}.$$

$C, O, R, L$ represent the portions of the state vector corresponding to common, opposite, right, and left respectively. The uppercase letters for $C$, $O$, $R$, $L$, $W$, $X$, $Y$, and $Z$ indicate that these are not necessarily scalars. For instance, for *Walker3D*, $O$ contains both $abdomen_x$ and $abdomen_z$. Similarly, the matrix, $(a_{ij})_{4\times4}$, is dimensionally consistent with the corresponding elements in the state vector. For example, $a_{2j}$ is a two-column wide block that matches with the two elements in $O$ for *Walker3D*. In addition, notice the negated $O$ and $X$, as well as the interchanged $R$ and $L$ are the effect of the mirroring functions. Overall, there are a total of 16 unknowns and 8 equations. A symmetric layer can be obtained by solving this system of equations. In particular, *NetAlt* contains symmetric layers of the following form,

$$(a_{ij})_{4\times4} = \begin{bmatrix} \alpha & 0 & \beta & \beta \\ 0 & \gamma & \beta & -\beta \\ \delta & \epsilon & \zeta & \eta \\ \delta & -\epsilon & \eta & \zeta \end{bmatrix}.$$

In order to maintain the symmetric policy constraint, the activation function applied to the negation portion, $O$, must be an odd function such as *tanh* or *softsign*. A similar procedure can be followed for intermediate and output layers, as long as the sizes for each of the portions are correctly maintained. Finally, a symmetric policy network can be constructed by stacking symmetric layers.

## A.3 Symmetry in DeepMimic Environment

To evaluate the effectiveness of phase-based mirroring, we ran an experiment for the original DeepMimic environment [Peng et al. 2018] in additional to the *Cassie* environment. In both cases, our data shows that phase-based mirroring does indeed make the learning faster. However, in the case of *DeepMimic*, the difference in final return is small between BASE and PHASE and only a minor difference can be seen from the video.