Enhancing Value Estimation Policies by Exploiting Symmetry in Agile Motion Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract: Motion planning tasks like catching, interception, and manipulation require high frequency perception and control to account for the agility required to complete the task. Reinforcement learning (RL) can produce such solutions, but can often be difficult to train and generalize. However, by exploiting the intrinsic geometric properties of agile task workspaces, we can enhance the performance of an RL and generalize it to new tasks. In this work we leverage geometric symmetry to enhance the performance of a value estimation policy (Actor Critic, A2C). Our method involves applying a geometric transformation to the observation during execution to provide the policy an alternate perspective of the current state. We show the effect of the symmetry exploitation policy on a trained A2C model on a WidowX reach task. The results show that by using symmetry exploitation, a trained model improves its performance, and generalizes to new tasks.

Keywords: Motion Planning, Reinforcement Learning

1 INTRODUCTION

Reinforcement learning (RL) is useful for motion planning tasks like robotic manipulation [1], juggling [2], and sports [3, 4], all of which require agility and high frequency control. However, the quality of the learned solution depends on the experience the model collects during training [5]. For example, a model that learns to control a robot arm may have higher value estimate and better performance with reaching areas to its left hand side rather than its right, despite the invariance of the model and task to reflections, e.g, the symmetry of the environment. Some methods rely on probabilistic sampling [6] to generalize for the symmetry, and others address it directly through experience augmentation [7] or latent space planning [8]. However, our intuition is to exploit this uncertainty in estimation as action alternatives. Considering the example above, we could present the learned model with the target in the left hand side and its symmetric position in the right hand side. If the model proposes different actions for either perspective, we choose the action leading to the higher reward by comparing the values of the states.

In this work, we propose a method to leverage the invariance of the work space, i.e. symmetry, to improve the performance of a value estimation method that requires no additional training. We demonstrate our method, the *symmetry exploitation policy*, on an actor critic model (A2C) trained on 3D WidowX robot manipulator environment [9], and find that using the policy improves performance and generalization to unseen tasks. This paper contributes the symmetry exploitation policy and evaluation for a simulated 3D robotic environment. with continuous actions and observations. We believe this method will help improve the performance of value estimation models that operate in symmetric environments, and help generalize them to unseen tasks.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

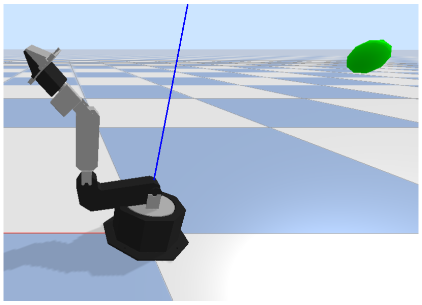


Figure 1: The WidowxReach-v28 environment, showing the WidowX robot arm with 6 joints, and the green random goal.

Algorithm 1 Symmetry Exploitation Policy

$Environment\ Initialization$
$o \leftarrow initial\ observation$
while *not done* **do**
  $o^m \leftarrow transform_{obs}(o)$
  if $V(o) \geq V(o^m)$ **then**
    $a \leftarrow \pi(o)$
  else
    $a^m \leftarrow \pi(o^m)$
    $a \leftarrow transform_{action}(a^m)$
  end if
  $o, done \leftarrow step(a)$
end while

2 Related Work

Geometric transformations are used in machine learning for data augmentation [10]. Commonly used in image processing, data augmentation can help generalization through increasing sample size or randomizing data. Such transformations include translations, rotations, reflections, and scaling [11, 12, 13]. Our method, on the other hand, operates in the motion planning domain. In the case of motion planning with reinforcement learning (RL), several existing works [14, 15, 16] augment input images with random textures and lighting conditions to generalize to real world observations. Closest to our work, Kostrikov et. al [7] uses geometric translation or shifting of input pixels to produce multiple Q values on which to regularize Q value estimation. In contrast, rather than using augmented observations to normalize or generalize, our method follows the maximum value actions and states as a policy, and does so during execution rather than training.

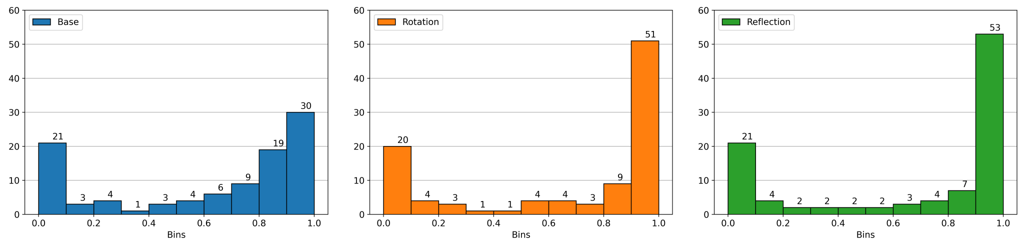
Towards a similar end, learning and planning in latent space, or state abstraction [17], also aims to bypass low level features of the environment by planning in a learned model of the task. In [18] an autoencoder learns to estimate latent space, where a sampling planner and dynamics estimator plan the policy. In a combined example, Srinivas et. al [8], use augmentation to learn high level features from pixel representations. These methods often have an encoder and decoder comparable to geometric transformations of our method. However, latent space planning relies on learning the latent space during training, whereas our method is applied during execution rather than training.

Outside of learning, symmetry has been extensively exploited for motion planning. Early work by Frazzoli [19], which led to more recent endeavors [20, 21], plan using motion primitives represented by equivalence classes defined by symmetries. Other work like that of Larkin et. al [22] reduce collision calculation by leveraging the symmetry of the geometric bodies. While our work shares the notion of symmetry exploitation, it specifically applies to estimated policies and values.

3 Method

Environment: We demonstrate our method in the WidowXReach-v28 environment as implemented by [9]. The environment, shown in Fig. 1, contains the WidowX robot, a fixed-base 6 degree of freedom manipulator arm. The robot's task is for its end-effector to reach a specific goal point in space, commonly referred to as a reach or fetch problem. Each episode is initialized with randomized goal location in the environment, and a fixed pose for the arm (with the base joint at angle 0). The reward is a negative value equal to the distance between the goal and the end-effector positions, until reaching a certain minimum threshold of 0.001 at which point the reward is $+1$. The state observation, o, is a 15-element vector containing the following: The first 9 indices are the relative x,y, and z coordinates between the end-effector and the base position, the relative coordinates between the end-effector and goal positions, and the absolute goal position. The last 6 indices are the 6 joint angles. The action is a vector of 6 values indicating changes of joint angles.

!]



(a) (b) (c)

Figure 2: Success rate of 100 models using the (a) base, (b) rotation, and (c) reflection policies.

Preliminary Training: We train 100 models with different starting weights by seeding each model with a different random number. The agents are trained for $30k$ episodes using the Advantage Actor Critic (A2C) [23], as in [9]. The network has two 64-node hidden dense layers. All models are trained on an NVIDIA Tesla V100 GPUs with Intel Xeon E-2146G @ 3.50 GHz.

Symmetry Exploitation Policy: The symmetry exploitation policy is a modification to value estimation policies during execution. It relies on inherent geometric invariance, e.g., symmetry, to augment the observation and provide the model with additional perspectives on the same state that might produce different actions. The policy decides which action to use based on the estimated value of the corresponding state. This process is described in $Algorithm 1$. During execution, we take the current observation of the state, $o \in O$, and apply the chosen geometric transformation, $transform_{obs}$, to get a transformed observation, o^m (line 4). The learned estimated value, $V : O \rightarrow R$, is then queried on both o and o^m (line 5). We compare the value estimate of both observations and choose the observation with higher value to feed into the learned policy, to get an action, a (lines 5 to 8). If the chosen observation was the transformed observation, o^m, we apply the inverse geometric transformation, $transform_{action}$, to the action, a^m (line 9).

In the case of the WidowX reach task, the workspace has both (1) a rotational symmetry about the z axis, along the xy-plane, due to the first joint of the robot arm being a rotation joint and (2) a reflective symmetry about the xz-plane along the y axis. For the reflective symmetry, $transform_{obs}$ negates the y-axis values of each coordinate component of the observation (o index 1,4, and 7) and the value of the first joint angle (o index 9). $transform_{action}$ negates the action for the first joint (a index 0). For the rotation symmetry, $transform_{obs}$ rotates the x,y,z values of each coordinate component of the observation (o indices 0-8), using angle θ. θ is equal to the current angle of the first joint (o index 0), which is set to 0 after the transform. Since the action is defined as relative angle changes, the action a^m requires no transformation for rotation symmetry.

4 Experiments & Results

We evaluate the symmetry exploitation policy compared to standard execution by preforming 100 episodes per policy per model, and measure the cumulative episode rewards and number of episodes in which the end effector reaches within 50mm of the goal. To note, despite the reward structure awarding the reach reward at distance of 1mm, all trained models fail to reach that range during execution. The remainder of the text will refer to the symmetry exploitation policy as *reflection* or *rotation* based on the geometric transformation used, and the standard execution policy as *base*.

Experiment 1: The first experiment shows the effect of the symmetry exploitation policy on the performance of the trained models. The results compare the performance of base policy to the rotation and reflection policies, with the success rates shown in Fig.2. The models achieve an average success rate of 61.50%, standard deviation ±37.64 with the base policy, 67.71% ±39.93 success with the rotation policy, and 66.64% ±40.02 success with the reflection policy. With the rotation policy, 73 of 100 models experience an increase in reward, and 89 achieve greater or equal success compared to the base policy. With the reflection policy, 91 models experience an increase in reward,

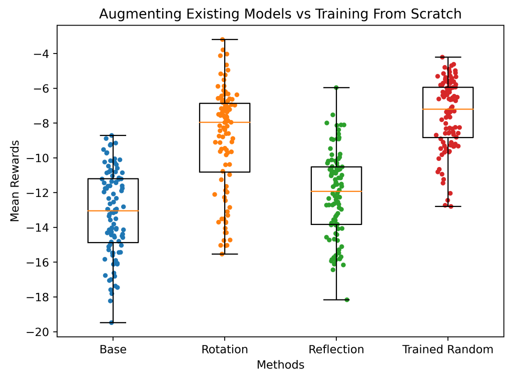


Figure 3: Mean reward of 100 models on an unseen task where the starting pose has a random base joint angle. The red dots represent 100 models that were trained on the new task.

and 84 achieve greater or equal success compared to base policy. From Fig. 2 we can see that population of models with low success rates remains unchanged by the symmetry exploitation policies. This is expected as the symmetry exploitation policy relies on the learned model to estimate the value and action, and if a model performs poorly there is little room for improvement. However, we can also see that the distribution of the other models shifts towards higher success rates. The base policy has 30 models with success above 90%, whereas both symmetry exploitation policies have over 50 of such models. These results show that the symmetry exploitation policy can be used to improve the performance of a trained model.

Experiment 2: The second experiment shows the effect of the symmetry exploitation policy in generalizing models to new tasks. We define a new task where the starting pose of the robot arm initializes with a random base joint angle. The results compare the performance of the base policy to the rotation and reflection policies. Additionally, as a comparison, we trained 100 new models on the task with the new parameters (referred to as Trained Random) and include their performance. As expected, the performance of the original models drops when given a new task, and the Trained Random models perform better since they were trained for this new task. However, when examining the mean episode rewards shown in Fig. 3, we can see that using the rotation policy shifts the distribution of the reward for the base models towards a similar distribution to that of the Trained Random models. By using symmetry exploitation, the models trained on the previous task are able to achieve rewards comparable to models trained on the new task. These results show that the symmetry exploitation policy helps a model to generalize to new tasks.

5 CONCLUSION

In this work we present a symmetry exploitation policy, which augments observations during execution through geometric transformations. We showcase the method on the WidowX reach task, a building block for many robotic tasks that require agile execution. The results show that the symmetry exploitation policy not only improves the performance of a trained model, but also allows it to generalize to new tasks and perform comparably to models trained on the new task. Since our method does not rely on special considerations during training as existing methods do, it can be applied post-hoc to any eligible learning method, conditioned on the symmetry of the environment. The task and its symmetries presented in this work are a proof of concept for the validity of the method. We have yet to study the effect of combining symmetries or using multiple alternative observations. Other considerations include defining symmetries implicitly and applying it to methods that don't estimate value. In future work, we plan on addressing these questions, in addition to applying this method to additional tasks, learning methods, and symmetries.

References

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html.

[2] K. Ploeger, M. Lutter, and J. Peters. High acceleration reinforcement learning for real-world juggling with binary rewards. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 642–653. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/ploeger21a.html.

[3] J. E. G. Baxter, T. Adamson, S. Sugaya, and L. Tapia. Exploring learning for intercepting projectiles with a robot-held stick. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 369–376, 2021. doi:10.1109/IROS51168.2021.9635993.

[4] T. Ding, L. Graesser, S. Abeyruwan, D. B. D'Ambrosio, A. Shankar, P. Sermanet, P. R. Sanketi, and C. Lynch. Goalseye: Learning high speed precision table tennis on a physical robot, 2022. URL https://arxiv.org/abs/2210.03662.

[5] V. Gudivada, A. Apon, and J. Ding. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20, 2017.

[6] P. Ladosz, L. Weng, M. Kim, and H. Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022. ISSN 1566-2535. doi:https://doi.org/10.1016/j.inffus.2022.03.003. URL https://www.sciencedirect.com/science/article/pii/S1566253522000288.

[7] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

[8] M. Laskin, A. Srinivas, and P. Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

[9] P. Aumjaud, D. McAuliffe, F. J. Rodríguez Lera, and P. Cardiff. rl_reach: Reproducible reinforcement learning experiments for robotic reaching tasks. *Software Impacts*, 8, 2021.

[10] K. T. Shorten, C. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:60, 2019.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017. ISSN 0001-0782. doi:10.1145/3065386. URL https://doi.org/10.1145/3065386.

[12] I. Sato, H. Nishimura, and K. Yokoi. Apac: Augmented pattern classification with neural networks, 2015. URL https://arxiv.org/abs/1505.03229.

[13] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012. doi:10.1109/CVPR.2012.6248110.

[14] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. In *Robotics: Science and Systems 2017*, 07 2017. doi:10.15607/RSS.2017.XIII.034.

[15] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi:10.1109/IROS.2017.8202133.

[16] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. 06 2018. doi:10.15607/RSS.2018.XIV.008.

[17] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker. Model-based reinforcement learning: A survey. URL `https://arxiv.org/abs/2006.16712`.

[18] B. Ichter and M. Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019. doi:10.1109/LRA.2019.2901898.

[19] E. Frazzoli. *Robust hybrid control for autonomous vehicle motion planning*. PhD thesis, Massachusetts Institute of Technology, 2001.

[20] K. Flaßkamp, S. Ober-Blöbaum, and K. Worthmann. Symmetry and motion primitives in model predictive control. *Mathematics of Control, Signals, and Systems*, 31(4):455–485, 2019.

[21] S. Ober-Blöbaum and S. Peitz. Explicit multiobjective model predictive control for nonlinear systems with symmetries. *International Journal of Robust and Nonlinear Control*, 31(2):380–403, oct 2020. doi:10.1002/rnc.5281. URL `https://doi.org/10.1002%2Frnc.5281`.

[22] N. Larkin, A. Short, Z. Pan, and S. Van Duin. Task space motion planning decomposition. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1688–1694, 2018. doi:10.1109/ICRA.2018.8460903.

[23] V. Konda and J. Tsitsiklis. Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, pages 1008–1014. MIT Press, 2000.