Integrating Symmetry into Differentiable Planning

Linfeng Zhao†, Xupeng Zhu†*, Lingzhi Kong†*, Robin Walters† and Lawson L.S. Wong†
†Khoury College of Computer Sciences
Northeastern University
Email: zhao.linf@northeastern.edu
*Equal Contribution

I. Introduction

Model-based planning usually struggles in complex problems, and planning in more structured and abstract space is a major solution [1, 2, 3, 4]. Symmetry is ubiquitous in learning and decision-making problems and can effectively reduce search space for planning. However, existing planning algorithms using symmetry assumes perfect dynamics knowledge, needs to explicitly build equivalence classes, or does not consider problem structure [5, 4, 6, 7, 8]. For example, if we use A* on path planning, we cannot specify visually obvious rotation symmetry in Figure 1, and need to detect in manually from the provided dynamics model. This would be even more challenging to detect in differentiable planning.

Nevertheless, symmetry in model-free deep reinforcement learning (RL) has been studied recently [9, 10]. However, it can only handle pixel-level "element-wise" symmetry, such as flipping or rotating state and action together. However, a critical benefit of model-free RL agents that enables great asymptotic performance is its end-to-end differentiability. This motivates us to combine the spirit of both: *is it possible to design an end-to-end differentiable planning algorithm that makes use of symmetry in environments?*

In this work, we propose to (1) avoid explicitly building equivalence classes for symmetric states while (2) realize planning in an end-to-end differentiable manner. We are motivated by work in the equivariant network and geometric deep learning community [11, 12, 13, 14, 15, 16], which treat an RGB image as a mapping $\mathbb{Z}^2 \to \mathbb{R}^3$ and apply equivariant convolutions between feature maps. It satisfies our desiderata: equivariant networks on images do not need to explicitly consider "symmetric pixels" while guarantee symmetry properties. Based on the intuition, we propose a framework, *Symmetric Planning* (SymPlan), to understand a straightforward but general problem, path planning, as operating like images, called steerable feature fields [14, 16]. We focus on 2D grid and prove that value iteration (VI) for 2D path planning is equivariant under the *isometries* of $\mathbb{Z}^2$: translations, rotations, and reflections, and further show that VI here is a special form of steerable convolution network [14]. This provides us a foundation to equip Value Iteration Network (VIN, [17]) with steerable convolution. We implement the equivariant steerable version of VIN, named SymVIN, and use a variant, GPPN, to build SymGPPN. Both SymPlan methods achieve great improvement on training efficiency and generalization performance to unseen random maps, which showcases the advantage of exploiting symmetry from environments for planning.

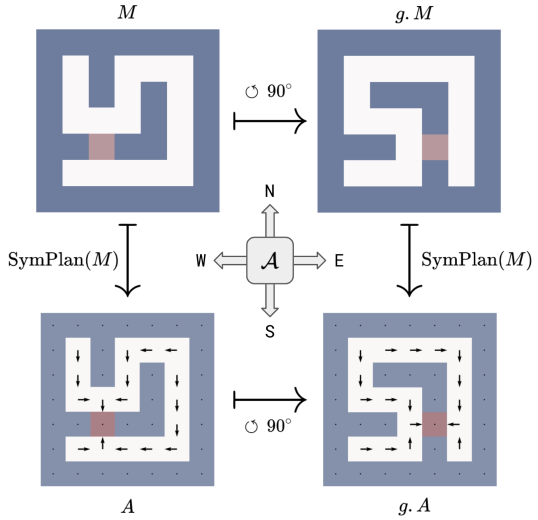

Figure 1: The *path planning problem* has symmetry, so we study how to *exploit* its symmetry in (differentiable) *planning*. Red dots are goal. The optimal actions $A = \texttt{SymPlan}(M)$ (bottom row) for the maps M (top row) are guaranteed to be equivariant $\texttt{SymPlan}(g.M) = g.\texttt{SymPlan}(M)$ under ↻ rotations for (2D) path planning. For example, the action in the NW corner of A is the same as the action in the SW corner of $g.A$, after also rotating the arrow ↻ 90°.

Our contributions are as follows:

- Understand the inherent symmetry in path planning problems (on 2D grids), formulate value iteration in as steerable convolution network, and connect both to incorporate symmetry into VI.
- Based on the formulation, implement equivariant steerable version of VIN and GPPN.
- Show significant improvement in training and generalization on 2D navigation and manipulation.

Our full version is available at https://arxiv.org/abs/2206.03674.

II. Symmetric Planning in Practice

In this section, we discuss how to achieve Symmetric Planning on 2D grids with E(2)-steerable CNNs [16]. We focus on implementing symmetric version of value iteration, SymVIN,

and generalize the methodology to make a symmetric version of a popular follow-up of VIN, GPPN [18].

Steerable value iteration. We have showed that, value iteration for path planning problems on $\mathbb{Z}^2$ consists of equivariant maps between steerable feature fields. It can be implemented as an equivariant steerable CNN, with recursively applying two alternating (equivariant) layers:

$$Q_k^a(s) = R_m^a(s) + \gamma \times [P_\theta^a \star V_k](s), V_{k+1}(s) = \max_a Q_k^a(s), \tag{1}$$

where $s \in \mathbb{Z}^2$, $k \in [K]$ indexes iteration, V_k, Q_k^a, R_m^a are steerable feature fields over $\mathbb{Z}^2$ output by equivariant layers, P_θ^a is a learned kernel in neural network, and $+, \times$ are element-wise operations.

Pipeline. We follow the pipeline in VIN [17]. The commutative diagram for the full pipeline is shown in Figure 2. The path planning task is given by a $m \times m$ spatial binary obstacle occupancy map and one-hot goal map, represented as a feature field $M : \mathbb{Z}^2 \to \{0, 1\}^2$. For the iterative process $Q_k^a \mapsto V_k \mapsto Q_{k+1}^a$, the reward field R_M is predicted from map M (by a 1×1 convolution layer) and the value field V_0 is initialized as zeros. The network output is (logits of) planned actions for all locations[1], represented as $A : \mathbb{Z}^2 \to \mathbb{R}^{|\mathcal{A}|}$, predicted from the final Q-value field Q_K (by another 1×1 convolution layer). The number of iterations K and the convolutional kernel size F of P_θ^a are set based on map size M, and the spatial dimension $m \times m$ is kept consistent.

Building Symmetric Value Iteration Networks. Given the pipeline of VIN fully on steerable feature fields, we are ready to build equivariant version with $\mathrm{E}(2)$-steerable CNNs [16]. The idea is to replace every `Conv2d` with a steerable convolution layer between steerable feature fields, and associate the fields with proper fiber representations $\rho(h)$.

VINs use ordinary CNNs and can choose the size of intermediate feature maps. The design choices in steerable CNNs is the feature fields and fiber representations (or *type*) for every layer [14, 16]. The main difference[2] in steerable CNNs is that we also need to tell the network how to *transform* every *feature field*, by specifying *fiber representations*, as shown in Figure 2.

Specification of input map and output action. We first specify *fiber representations* for the input and output field of the network: map M and action A. For input **occupancy map and goal** $M : \mathbb{Z}^2 \to \{0, 1\}^2$, it does not D_4 to act on the 2 channels, so we use two copies of trivial representations $\rho_M = \rho_{\text{triv}} \oplus \rho_{\text{triv}}$. For **action**, the final action output $A : \mathbb{Z}^2 \to \mathbb{R}^{|\mathcal{A}|}$ is for logits of four actions $\mathcal{A} =$ `(north, west, south, east)` for every location. If we use $H = C_4$, it naturally acts on the four actions (ordered $\circlearrowleft$) by *cyclically* $\circlearrowleft$ *permuting* the $\mathbb{R}^4$ channels. However, since the D_4 group has 8 elements, we need a *quotient representation*, see [16].

[1]Technically, it also includes values or actions for obstacles, since the network needs to learn to approximate the reward $R_M(s, \Delta s) = -\infty$ with enough small reward and avoid obstacles.

Specification of intermediate fields: value and reward. Then, for the intermediate feature fields: Q-values Q_k, state value V_k, and reward R_m, we are free to choose fiber representations, as well as the width (number of copies). For example, if we want 2 copies of regular representation of D_4, the feature field has $2 \times 8 = 16$ channels and the stacked representation is 16×16 (by direct-sum).

For the Q**-value field** $Q_k^a(s)$, we use representation ρ_Q and its size as C_Q. We need at least $C_A \geq |\mathcal{A}|$ channels for all actions of $Q(s, a)$ as in VIN and GPPN, then stacked together and denoted as $Q_k \triangleq \bigoplus_a Q_k^a$ with dimension $Q_k : \mathbb{Z}^2 \to \mathbb{R}^{C_Q * C_A}$. Therefore, the representation is direct-sum $\bigoplus \rho_Q$ for C_A copies. The **reward** is implemented similarly as $R_M \triangleq \bigoplus_a R_M^a$ and must have same dimension and representation to add element-wisely. For **state value** field, we denote the choose as fiber representation as ρ_V and its size C_V. It has size $V_k : \mathbb{Z}^2 \to \mathbb{R}^{C_V}$ Thus, the steerable kernel is *matrix-valued* with dimension $P_\theta : \mathbb{Z}^2 \to \mathbb{R}^{(C_Q * C_A) \times C_V}$. In practice, we found using *regular representations* for all three works the best. It can be viewed as "augmented" state and is related to group convolution.

III. Experiments

We experiment VIN, GPPN and our SymPlan methods on given maps.

Environments and datasets. We demonstrate the idea in two major robotics tasks: *navigation* and *manipulation*. We focus on the 2D regular grid setting for path planning, as adopted in prior work [17, 18, 19]. For each task, we consider using *given* maps (2D navigation and 2-DOF configuration-space manipulation). In the latter case, the planner needs to jointly learn a mapper that converts egocentric panoramic images (visual navigation) or workspace states (workspace manipulation) into plannable loss, as in [18, 19]. In both cases, we randomly generate training, validation and test data of $10K/2K/2K$ maps for all map sizes, to demonstrate data efficiency and generalization ability of symmetric planning. Note that the test maps are unlikely to be symmetric to the training maps by any transformation from the symmetry groups G. For all environments, the planning domain is the 2D regular grid $\mathcal{S} = \Omega = \mathbb{Z}^2$, and the action space is to move in 4 $\circlearrowleft$ directions[3]: $\mathcal{A} =$ `(north, west, south, east)`.

Methods: planner networks. We compare five planner methods, where two are our SymPlan version of their non-equivariant counterparts. Our equivariant implementation is based on *Value Iteration Networks* (**VIN**, [17]) and *Gated Path Planning Networks* (**GPPN**, [18]). We implement the equivariant version of VIN, named **SymVIN**. For GPPN, we first obtained a *fully convolutional* version, named **ConvGPPN** [Redacted for anonymous review], and furthermore **SymGPPN** with steerable CNNs. All methods use (equivariant) convolutions with *circular padding* in planning in

[3]Note that the MDP action space $\mathcal{A}$ needs to be *compatible* with the group action $G \times \mathcal{A} \to \mathcal{A}$. Since the E2CNN package [16] uses *counterclockwise* rotations $\circlearrowleft$ as generators for rotation groups C_n, the action space needs to be *counterclockwise* $\circlearrowleft$.

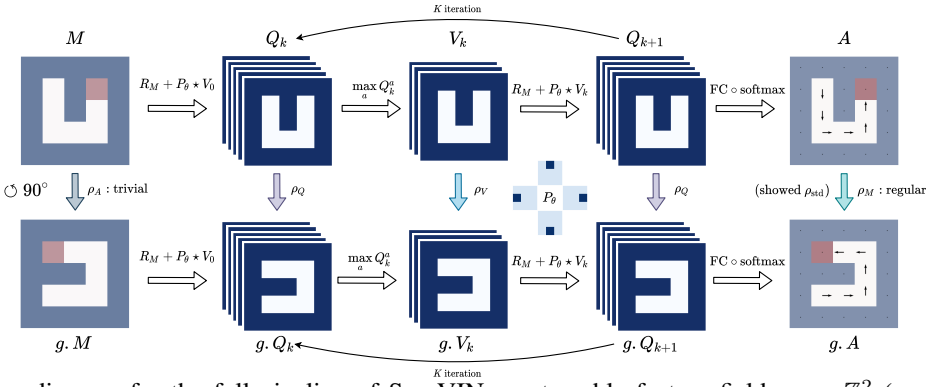

Figure 2: Commutative diagram for the full pipeline of SymVIN on steerable feature fields over $\mathbb{Z}^2$ (every grid). If rotating the input map M by $\pi_M(g)$ of any g, the output action $A = \texttt{SymVIN}(M)$ is guaranteed to be transformed by $\pi_A(g)$, i.e. the entire steerable SymVIN is equivariant under induced representations π_M and π_A: $\texttt{SymVIN}(\pi_M(g)M) = \pi_A(g)\texttt{SymVIN}(M)$. We use stacked feature fields to emphasize that SymVIN supports direct-sum of representations beyond scalar-valued.

Figure 3: **(Left)** A visual navigation environment rendered from a randomly generated 7×7 maze **(Middle)**, where the hover is the visualization of four views at position $(5, 3)$. **(Right)** A 2-joint manipulation task in workspace (topdown) and configuration space (2 DOFs) in 18×18 resolution.

Table I: Averaged test success rate (%).

Method (10K Data)	Navigation 15×15	Navigation 28×28	Navigation 50×50	Manipulation 18×18	Manipulation 36×36
VIN	66.97	67.57	57.92	77.82	84.32
SymVIN	98.99	98.14	86.20	99.98	99.36
GPPN	96.36	95.77	91.84	2.62	1.68
ConvGPPN	99.75	99.09	97.21	99.98	99.95
SymGPPN	**99.98**	**99.86**	**99.49**	**100.00**	**99.99**

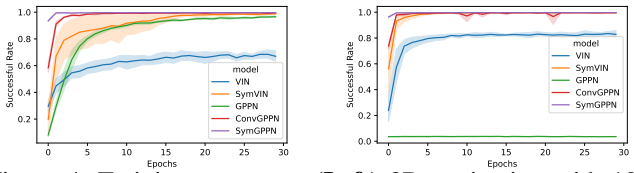

Figure 4: Training curves on **(Left)** 2D navigation with 10K of 15×15 maps and on **(Right)** 2DoFs manipulation with 10K of 18×18 maps in configuration space. Faded areas indicate standard error.

configuration spaces for the manipulation tasks, except GPPN that is not fully convolutional. Chaplot et al. [19] propose SPT based on Transformers, while integrating symmetry to Transformers is beyond steerable convolutions, thus we do not consider it but still adopt some useful setup.

Training and evaluation. We report success rate and training curves over 3 seeds. The training process (on given maps) follows [17, 18], where we train 30 epochs with batch size 32, and use kernel size $F = 3$ by default. The gradient clip threshold is set to 5. The default batch size is 32, while we need to reduce for some GPPN variants, since LSTM consumes much more memory.

A. Planning on given maps

Environmental setup. In the **2D navigation** task, the map and goal are randomly generated, where the map size is $\{15, 28, 50\}$. In **2-DOF manipulation** in configuration space, we adopt the setting in [19] and train networks to take as input of configuration space, represented by two joints. We randomly generate 0 to 5 obstacles in the manipulator workspace. Then the 2 degree-of-freedom (DOF) configuration space is constructed from workspace and discretized into 2D grid with sizes $\{18, 36\}$, corresponding to bins of $20°$ and $10°$, respectively. All methods are trained using the same network size, where for equivariant versions, we use *regular* representations for all layers, which has size $|D_4| = 8$. We keep the same parameters for all methods, so all equivariant convolution layers with *regular* representations will have higher embedding sizes. Due to memory constraint, we use $K = 30$ iterations for 2D maze navigation, and $K = 27$ for manipulation. We use kernel sizes $F = \{3, 5, 5\}$ for $m = \{15, 28, 50\}$ navigation, and $F = \{3, 5\}$ for $m = \{18, 36\}$ manipulation.

Results. We show the averaged test results for both 2D navigation and C-space manipulation tasks on generalizing to unseen maps (Table I) and the training curves for all methods (Figure 4). For VIN series, our SymVIN is much better than the vanilla VIN in terms of generalization and training performance in both environments, which learns much faster and achieves almost perfect asymptotic performance. As for GPPN, we found the fully convolutional variant ConvGPPN actually works better than the original one in [18], especially in learning speed. However, SymVIN does fluctuate in some runs, which seems to come from initialization and label. SymGPPN further boosts ConvGPPN and outperforms all other methods. One exception is GPPN learns poorly in C-space manipulation. For GPPN, the added circular padding in the convolution encoder leads to gradient vanishing problem.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

[2] Lihong Li, Thomas J. Walsh, and M. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *AI&M*, 2006.

[3] Balaraman Ravindran and Andrew G Barto. *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst, 2004.

[4] Maria Fox and Derek Long. Extending the exploitation of symmetries in planning. In *In Proceedings of AIPS'02*, pages 83–91, 2002.

[5] Maria Fox and Derek Long. The Detection and Exploitation of Symmetry in Planning Problems. In *In IJCAI*, pages 956–961. Morgan Kaufmann, 1999.

[6] Nir Pochter, Aviv Zohar, and Jeffrey S. Rosenschein. Exploiting Problem Symmetries in State-Based Planners. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, August 2011. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3732.

[7] Martin Zinkevich and Tucker Balch. Symmetry in Markov decision processes and its implications for single agent and multi agent learning. In *In Proceedings of the 18th International Conference on Machine Learning*, pages 632–640. Morgan Kaufmann, 2001.

[8] Shravan Matthur Narayanamurthy and Balaraman Ravindran. On the hardness of finding symmetries in Markov decision processes. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 688–695, Helsinki, Finland, 2008. ACM Press. ISBN 978-1-60558-205-4. doi: 10/bkswc2. URL http://portal.acm.org/citation.cfm?doid=1390156.1390243.

[9] Elise van der Pol, Daniel E. Worrall, Herke van Hoof, Frans A. Oliehoek, and Max Welling. MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning. *arXiv:2006.16908 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.16908. arXiv: 2006.16908.

[10] Dian Wang, Robin Walters, and Robert Platt. $\mathrm{SO}(2)$-Equivariant Reinforcement Learning. September 2021. URL https://openreview.net/forum?id=7F9cOhdvfk_.

[11] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[12] Taco Cohen, Mario Geiger, and Maurice Weiler. A General Theory of Equivariant CNNs on Homogeneous Spaces. *arXiv:1811.02017 [cs, stat]*, January 2020. URL http://arxiv.org/abs/1811.02017. arXiv: 1811.02017.

[13] Risi Kondor and Shubhendu Trivedi. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. *arXiv:1802.03690 [cs, stat]*, November 2018. URL http://arxiv.org/abs/1802.03690. arXiv: 1802.03690.

[14] Taco S. Cohen and Max Welling. Steerable CNNs. November 2016. URL https://openreview.net/forum?id=rJQKYt5ll.

[15] Taco S. Cohen and Max Welling. Group Equivariant Convolutional Networks. *arXiv:1602.07576 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1602.07576. arXiv: 1602.07576.

[16] Maurice Weiler and Gabriele Cesa. General $E(2)$-Equivariant Steerable CNNs. *arXiv:1911.08251 [cs, eess]*, April 2021. URL http://arxiv.org/abs/1911.08251. arXiv: 1911.08251.

[17] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4949–4953, Melbourne, Australia, August 2017. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi: 10/ggjfst. URL https://www.ijcai.org/proceedings/2017/700.

[18] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. Gated Path Planning Networks. *arXiv:1806.06408 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.06408. arXiv: 1806.06408.

[19] Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. Differentiable Spatial Planning using Transformers. *arXiv:2112.01010 [cs]*, December 2021. URL http://arxiv.org/abs/2112.01010. arXiv: 2112.01010.