PyPose: A Library for Robot Learning with Physics-based Optimization
Supplementary Material

Chen Wang[1,2,✉], Dasong Gao[1,3], Kuan Xu[4], Junyi Geng[1], Yaoyu Hu[1], Yuheng Qiu[1], Bowen Li[1], Fan Yang[5], Brady Moon[1], Abhinav Pandey[6], Aryan[1,7], Jiahe Xu[1], Tianhao Wu[8], Haonan He[1], Daning Huang[6], Zhongqiang Ren[1], Shibo Zhao[1], Taimeng Fu[9], Pranay Reddy[10], Xiao Lin[11], Wenshan Wang[1], Jingnan Shi[3], Rajat Talak[3], Kun Cao[4], Yi Du[2], Han Wang[4], Huai Yu[12], Shanzhao Wang[13], Siyu Chen[4], Ananth Kashyap[14], Rohan Bandaru[15], Karthik Dantu[2], Jiajun Wu[16], Lihua Xie[4], Luca Carlone[3], Marco Hutter[5], Sebastian Scherer[1]
https://pypose.org

A. Sample Code of `LieTensor`

The following code sample shows how to rotate random points and compute the gradient of batched rotation.

```
>>> import torch, pypose as pp

>>> # A random so(3) LieTensor
>>> r = pp.randn_so3(2, requires_grad=True)
so3Type LieTensor:
tensor([[ 0.1606,  0.0232, -1.5516],
        [-0.0807, -0.7184, -0.1102]],
        requires_grad=True)

>>> R = r.Exp() # Equivalent to: R = pp.Exp(r)
SO3Type LieTensor:
tensor([[ 0.0724,  0.0104, -0.6995,  0.7109],
        [-0.0395, -0.3513, -0.0539,  0.9339]],
        grad_fn=<AliasBackward0>)

>>> p = R @ torch.randn(3) # Rotate random point
tensor([[ 0.8045, -0.8555,  0.5260],
        [ 0.3502,  0.8337,  0.9154]],
        grad_fn=<ViewBackward0>)

>>> p.sum().backward()      # Compute gradient
>>> r.grad                  # Print gradient
tensor([[-0.7920, -0.9510,  1.7110],
        [-0.2659,  0.5709, -0.3855]])
```

✉Corresponding Author. chenwang@dr.com
[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.
[2]State University of New York at Buffalo, NY 14260, USA.
[3]Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[4]Nanyang Technological University, Singapore 639798.
[5]ETH Zürich, 8092 Zürich, Switzerland.
[6]Pennsylvania State University, University Park, PA 16801, USA.
[7]Delhi Technological University, Delhi, India.
[8]University of Virginia, Charlottesville, VA 22904, USA.
[9]The Chinese University of Hong Kong, Shenzhen, China.
[10]University of Massachusetts Amherst, MA 01003, USA.
[11]Georgia Institute of Technology, Atlanta, GA 30332, USA
[12]Wuhan University, Hubei 430072, China.
[13]University of Michigan, Ann Arbor, MI 48109, USA.
[14]Fox Chapel Area High School, Pittsburgh, PA 15238, USA.
[15]Lexington High School, Lexington, MA 02421, USA.
[16]Stanford University, Stanford, CA 94305, USA.

B. Sample Code of `Optimizer`

We show how to estimate batched transform inverse by a 2nd-order optimizer. Two usage options for a `scheduler` are provided, each of which can work independently.

```
>>> from torch import nn
>>> import torch, pypose as pp
>>> from pypose.optim import LM
>>> from pypose.optim.strategy import Constant
>>> from pypose.optim.scheduler \
    import StopOnPlateau

>>> class InvNet(nn.Module):

        def __init__(self, *dim):
            super().__init__()
            init = pp.randn_SE3(*dim)
            self.pose = pp.Parameter(init)

        def forward(self, input):
            error = (self.pose @ input).Log()
            return error.tensor()

>>> device = torch.device("cuda")
>>> input = pp.randn_SE3(2, 2, device=device)
>>> invnet = InvNet(2, 2).to(device)
>>> strategy = Constant(damping=1e-4)
>>> optimizer = LM(invnet, strategy=strategy)
>>> scheduler = StopOnPlateau(optimizer,
                              steps=10,
                              patience=3,
                              decreasing=1e-3,
                              verbose=True)

>>> # 1st option, full optimization
>>> scheduler.optimize(input=input)

>>> # 2nd option, step optimization
>>> while scheduler.continual():
        loss = optimizer.step(input)
        scheduler.step(loss)
```