

pykitPIV: PyTorch-compatible synthetic image generation for training flow estimation algorithms in particle image velocimetry

Kamila Zdybał^{*a}, Claudio Mucignat^a and Ivan Lunati^a

^aLaboratory for Computational Engineering, Swiss Federal Laboratories for Materials Science and Technology, Empa, Dübendorf, Switzerland

ARTICLE INFO

Keywords:

particle image velocimetry; flow estimation; convolutional neural networks; Python

ABSTRACT

We describe `pykitPIV`, a PyTorch-compatible Python library for generating synthetic images that mimic those obtained from particle image velocimetry (PIV) experiments. The library can be readily ported to training machine learning algorithms, such as convolutional neural networks (CNNs), or reinforcement learning (RL). Our image generation exploits the kinematic relationship between two PIV snapshots and advects particles from one time frame at t_1 , to the next at $t_1 + \Delta t$, with a second-order accurate numerical scheme. This results in paired image intensities, I_1 and I_2 that are separated by Δt in time. The goal of this library is to give the user a lot of flexibility in selecting various parameters that would normally be available in an experimental setting such as particle seeding density, thickness of the laser plane, camera exposure, particle loss due to out-of-plane movement, or Δt between images. The richness of image generation can help answer outstanding questions in training CNNs or RL agents.

1. Introduction

The last decade has seen many advances in training convolutional neural networks (CNNs) for flow estimation, *i.e.*, predicting motion information from consecutive static image frames. To date, numerous interesting network architectures have been developed. This includes various implementations of FlowNets [1–3], spatial pyramid network (SPyNet) [4], pyramid, warping and cost-volume network (PWC-Net) [5], and recurrent all-pairs field transforms (RAFT) [6]. In addition, the idea of the iterative residual refinement (IRR) [7] allowed for significant reduction in the number of trainable parameters thanks to weight-sharing at several levels of successively upscaling image resolution.

Particle image velocimetry (PIV) can especially profit from those architectures, since its main goal is to predict flow targets, such as velocity components, velocity magnitude, or vorticity, from paired snapshots of illuminated tracer particles injected into the flow. Recently, RAFT-PIV [8] and Lightweight Image-Matching Architecture (LIMA) [9] were proposed as versions of CNNs that are specifically optimized for predicting flow targets from velocimetry experiments. Thanks to this targeted optimization of CNN architecture and training parameters, RAFT-PIV achieves high accuracy and LIMA is significantly lighter than its predecessors.

To advance the accuracy of training CNNs for complex PIV applications, a number of research questions will have to be addressed next:

1. How rich does the training dataset need to be to remain applicable in a given experimental setting?
2. What degree of data augmentation is sufficient to transfer knowledge from one trained CNN to the next when changing experimental settings?

3. At what time separation, Δt , would the current CNN architectures fail?

To help researchers answer those questions, we propose the present Python library `pykitPIV` – **Python kinematic training for PIV**. Our library is compatible with PyTorch [10, 11] to allow for easier porting with machine learning algorithms.

2. Software overview

All functionalities of `pykitPIV` are organized in five classes. Fig. 1 illustrates the hierarchy of using `pykitPIV` classes and briefly describes what can be achieved with each class.

2.1. Class: Particle

The `Particle` class seeds the two-dimensional flow domain with particles. The user can steer the range of particle diameters and the seeding density.


2.2. Class: FlowField

The `FlowField` class allows to generate the velocity field to be applied on the two-dimensional domain. We implemented several methods to generate velocity fields, such as random smooth field, checkered field, Chebyshev polynomial field, or simple harmonics field. Those are illustratively visualized in Fig. ??a-d. This variety of velocity fields span cases with smooth and sharp velocity gradients and can help put machine learning algorithms to test.

The user also has the option of uploading an external velocity field, *e.g.*, coming from numerical simulation of Navier-Stokes equations (*cf.* Fig. ??e), or coming from a synthetic turbulence generator (*cf.* Fig. ??f).

2.3. Class: Motion

The `Motion` class applies the flow field to the particles. It uses RK-4 numeric scheme to advect particles by a user-

 kamila.zdybal@gmail.com (K. Zdybał*)
ORCID(s):

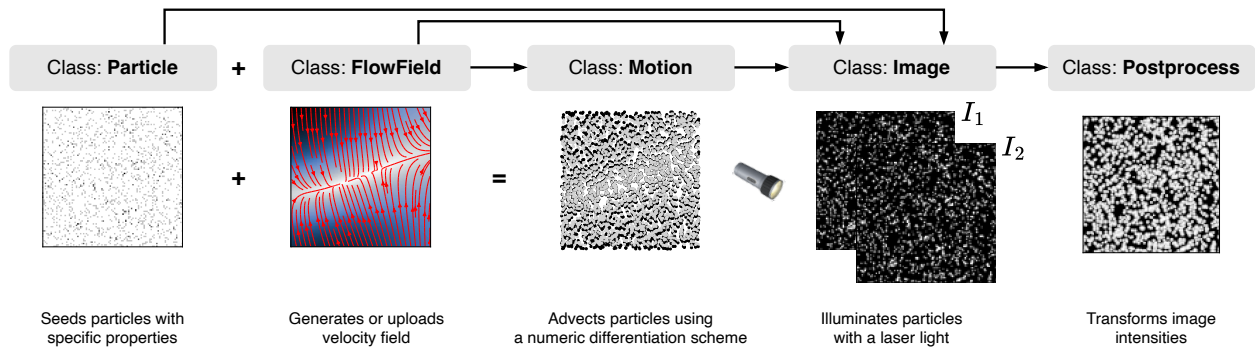


Figure 1: Order of using `pykitPIV` classes. At each stage of synthetic image generation, the user has freedom in selecting various parameters that would normally be available in an experimental setting such as particle seeding density, thickness of the laser plane, camera exposure, particle loss due to out-of-plane movement, or time separation between images.

specified time separation, Δt . The main output of this class are paired PIV images, I_1 and I_2 .

2.4. Class: Image

The `Image` class helps generate image intensities. It adds the reflected laser light to the generated image pairs. The core functionality is to add Gaussian intensity to each particle. The user has a lot of flexibility in setting up the laser plane and camera properties. The user can also steer the amount of particles lost between frame I_1 and I_2 due to out-of-plane movement.

The PIV image pair tensor has shape $(N, 2, H, W)$, where N is the batch size, H is image height and W is image width. The second dimension can be thought of as the number of channels and those correspond to I_1 and I_2 , respectively. This is compatible with tensor shape accepted by convolutional layers implemented in PyTorch. Note that the user can generate a whole batch, N , of images all at once.

2.5. Class: Postprocess

The `Postprocess` class contains functions that apply transformations to generated images. It can be especially useful for data augmentation.

3. Discussion

4. Conclusions

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author contributions

Acknowledgments

References

- [1] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, FlowNet: Learning optical flow with convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 2758–2766.
- [2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of optical flow estimation with deep networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2462–2470.
- [3] T.-W. Hui, X. Tang, C. C. Loy, Liteflownet: A lightweight convolutional neural network for optical flow estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8981–8989.
- [4] A. Ranjan, M. J. Black, Optical flow estimation using a spatial pyramid network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4161–4170.
- [5] D. Sun, X. Yang, M.-Y. Liu, J. Kautz, PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8934–8943.
- [6] Z. Teed, J. Deng, Raft: Recurrent all-pairs field transforms for optical flow, in: Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, Springer, 2020, pp. 402–419.
- [7] J. Hur, S. Roth, Iterative residual refinement for joint optical flow and occlusion estimation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 5754–5763.
- [8] C. Lagemann, K. Lagemann, S. Mukherjee, W. Schröder, Deep recurrent optical flow learning for particle image velocimetry data, *Nature Machine Intelligence* 3 (7) (2021) 641–651.
- [9] L. Manickathan, C. Mucignat, I. Lunati, A lightweight neural network designed for fluid velocimetry, *Experiments in Fluids* 64 (10) (2023) 161.

- [10] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).
- [11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, *Advances in neural information processing systems* 32 (2019).