

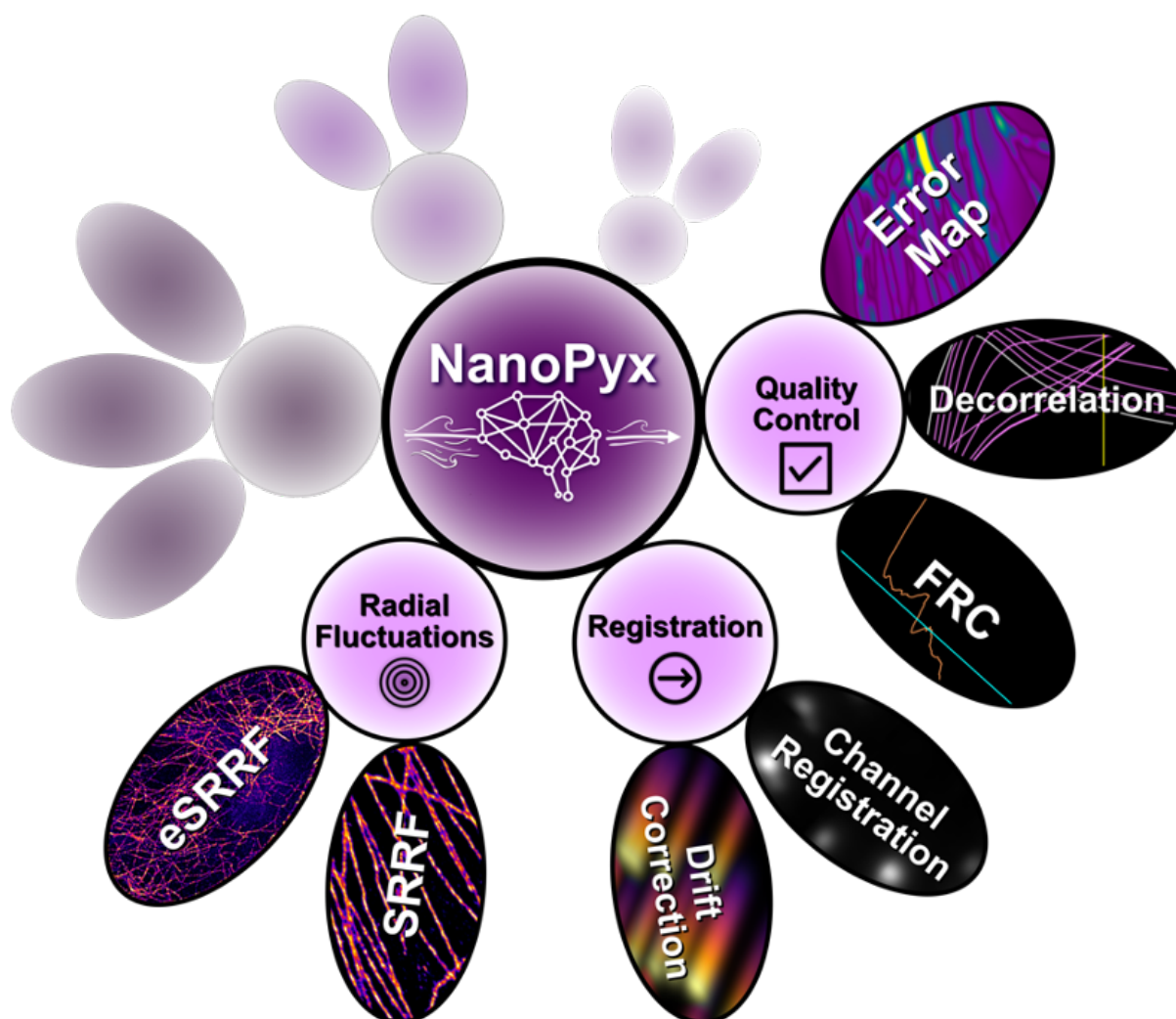
NanoPyx User Manual

Table of Contents

What is NanoPyx?	3
Installation	4
Usage	5
Methods	5
References	7

What is NanoPyx?

NanoPyx is a Python library specialized in the analysis of light microscopy and super-resolution data. It is a successor to NanoJ, which is a Java library for the analysis of super-resolution microscopy data.



NanoPyx focuses on performance, powered by the Liquid Engine. It implements methods for the bioimage analysis field, with a special emphasis on those developed by the Henriques Laboratory. It is distributed as a Python Library, as Codeless Jupyter Notebook (that can be run locally or on Google Colab) and as a napari plugin.

In this guide we briefly describe installation of the Python library and a small demo on how to use it. We also provide a small description of the currently implemented methods and workflows.

Online resources:

- Github repository - <https://github.com/HenriquesLab/NanoPyx>
- Documentation - <https://henriqueslab.github.io/NanoPyx/nanopyx.html>
- bioRxiv preprint - <https://www.biorxiv.org/content/10.1101/2023.08.13.553080v1>

- “What is NanoPyx?” – <https://youtu.be/iAdgusBAU0Q?si=UPxKXqdpJBeT53Gr>
- “How to use NanoPyx in Google Colab?” - <https://youtu.be/KD0RzoiFnd4?si=y8EfB8bdP7rTQOb8>
- “Managing Scientific Python environments using Conda, Mamba and friends” by Robert Haase - <https://focalplane.biologists.com/2022/12/08/managing-scientific-python-environments-using-conda-mamba-and-friends/>

Installation

NanoPyx is compatible and was tested with Python 3.9, 3.10, 3.11 in MacOS, Windows and Linux. Installation time depends on your hardware and internet connection but should not take more than 5 minutes.

We recommend installation of the NanoPyx Python library through pip:

```
pip install nanopyx
```

If you want to install with support for Jupyter notebooks:

```
pip install nanopyx[jupyter]
```

or if you want to install with all optional dependencies:

```
pip install nanopyx[all]
```

To install latest development version:

```
pip install git+https://github.com/HenriquesLab/NanoPyx.git
```

To install our napari plugin:

```
pip install napari-nanopyx
```

NOTE:

If you wish to compile the NanoPyx library from source on a MacOS computer, you will need to install the following dependencies:

- Homebrew from <https://brew.sh>

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- gcc, llvm and libomp from Homebrew through the command:

```
brew install gcc llvm libomp
```

Usage

NanoPyx can be used as a Python library and included as part of your Python workflows. It is also distributed as Codeless Jupyter notebooks that can be run locally or through Google Colab. To run the notebooks locally you will need a local Python installation. Then, install NanoPyx along with its notebook requirements:

```
pip install nanopyx[jupyter]
```

The notebooks are built to provide access to a specific bioimage analysis method workflow with minimal code exposure. You can either choose to use the provided example data or upload your own dataset. Follow the steps documented in each notebook to run the entire workflow and if you need additional help or want to see an example being run check out our video tutorial.

<https://www.youtube.com/watch?v=KD0RzofNd4>

We also developed a napari plugin that implements all NanoPyx methods as a single package. It is installed separately but requires an installation of NanoPyx.

```
pip install napari-nanopyx
```

Methods

Image registration

NanoPyx makes available 2D drift correction and channel registration. Both are based on using phase correlation to find the best match possible between frames/channels [1].

Drift Correction Parameters:

Requires an image stack with shape: (time, rows, columns)

- Reference Frame: Which frame to be used as reference. Either always use the first frame (better for fixed cells) or the previous frame (better for live cells).
- Max Expected Drift: Maximum amount of expected drift in pixels.
- Time Averaging: Whether to register each individual frame, if using 1, or how many frames to average before calculating drift correction (better for single molecule data). Output keeps the original number of frames; single frame drift estimation is calculated by interpolating using the calculated drift of the averaged image stack.

Channel Registration Parameters:

Requires an image stack with shape: (channel, rows, columns).

- Reference Channel: Which channel to be used as reference.
- Max Expected Shift: Maximum amount of expected shift between channels, in pixels.
- Blocks per Axis: As channel misalignment is not always homogeneous across the field of view, shift can be calculated for individual blocks of the field of view. This parameter sets how many blocks are created along both axes.

- Minimum Similarity: Since smaller blocks may lead to shift calculation in areas of the image without any cells, minimum similarity can be used to define the minimum Pearson's Correlation Coefficient, between two blocks of different channels, required to use the calculated shifts as part of the registration.

SRRF

Currently NanoPyx allows users to generate super-resolved images using SRRF [2].

SRRF Parameters:

Requires an image stack with shape: (time, rows, columns)

- Ring Radius: Radius of the ring used to calculate the radially image.
- Magnification: Desired magnification for the generated radially image.
- SRRF order: Flag for types of SRRF temporal correlations. Order = -1: pairwise product sum; Order = 0: maximum intensity projection; Order = 1: mean; Order = 2,3 or 4: autocorrelation function of order 2, 3 or 4.
- Frames-per-timepoint: How many frames of the original image stack are used to calculate a single SRRF frame. For example, given an input image with 500 frames, if using 100 frames per timepoint, SRRF will generate an image stack with 5 super-resolved frames.

eSRRF

NanoPyx also implements eSRRF (enhanced Super-Resolution Radial Fluctuations) which is an extension of the SRRF method [6].

eSRRF Parameters:

Requires an image stack with shape: (time, rows, columns)

- Ring Radius: radius of for the radial gradient convergence (RGC)
- Sensitivity: sensitivity of the RGC
- Magnification: Desired magnification for the generated RGC image.
- eSRRF order: Flag for types of eSRRF temporal correlations. Str = "AVG" – average; Str = "VAR" – variance; Str = "TAC2" – 2nd order autocorrelation.
- Frames-per-timepoint: How many frames of the original image stack are used to calculate a single eSRRF frame. For example, given an input image with 500 frames, if using 100 frames per timepoint, eSRRF will generate an image stack with 5 super-resolved frames.

Quality Control

NanoPyx implements the methods available in NanoJ-SQUIRREL [3] (Error Map and FRC [4]), as well as Image Decorrelation Analysis [5]

FRC Parameters:

Requires an image stack with shape: (slices, rows, columns)

- Pixel Size: Pixel size of the image. Used to calculate resolution values.
- Units: Pixel size units.

- First/Second Frame: As FRC is calculated between two frames of the same image stack, these parameters determine which two frames are used for the calculation.

Image Decorrelation Analysis Parameters:

Requires an image stack with shape: (slices, rows, columns)

- Pixel Size: Pixel size of the image. Used to calculate resolution values.
- Units: Pixel size units.
- Radius Min/Max: Resolution calculation by Decorrelation Analysis is performed in the frequency space. These parameters define the range of radii to be used in the calculation.

References

- [1] R. F. Laine *et al.*, 'NanoJ: a high-performance open-source super-resolution microscopy toolbox', *J. Phys. Appl. Phys.*, vol. 52, no. 16, p. 163001, Feb. 2019, doi: 10.1088/1361-6463/ab0261.
- [2] S. Culley, K. L. Tosheva, P. M. Pereira, and R. Henriques, 'SRRF: Universal live-cell super-resolution microscopy', *Int. J. Biochem. Cell Biol.*, vol. 101, pp. 74–79, 2018, doi: <https://doi.org/10.1016/j.biocel.2018.05.014>.
- [3] S. Culley *et al.*, 'Quantitative mapping and minimization of super-resolution optical imaging artifacts', *Nat. Methods*, vol. 15, no. 4, pp. 263–266, Apr. 2018, doi: 10.1038/nmeth.4605.
- [4] R. P. J. Nieuwenhuizen *et al.*, 'Measuring image resolution in optical nanoscopy', *Nat. Methods*, vol. 10, no. 6, pp. 557–562, Jun. 2013, doi: 10.1038/nmeth.2448.
- [5] A. Descloux, K. S. Großmayer, and A. Radenovic, 'Parameter-free image resolution estimation based on decorrelation analysis', *Nat. Methods*, vol. 16, no. 9, pp. 918–924, Sep. 2019, doi: 10.1038/s41592-019-0515-7.
- [6] Romain F. Laine *et al.*, 'High-fidelity 3D live-cell nanoscopy through data-driven enhanced super-resolution radial fluctuation', *bioRxiv*, p. 2022.04.07.487490, Jan. 2022, doi: 10.1101/2022.04.07.487490.