Visualization of Multi-Dimensional Data – The data-slicer Package

Kevin Kramer[1] **and Johan Chang**[1]

1 Physik Institut, Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland

DOI: 10.21105/joss.02969

Software

- Review
- Repository
- Archive

Editor: Tim Tröndle

Reviewers:

- @sabinomaggi
- @Chilipp

Submitted: 16 December 2020
Published: 15 April 2021

Statement of Need

From prehistoric cave-wall paintings to the invention of print and most recently electronic hard-disks, human data storage capacity has evolved tremendously. Information/data is of great value and hence associated with innovation and technological progress. This is especially true in analytical disciplines i.e. all sciences ranging from physics to psychology and medicine. In observational sciences, most measurement techniques undergo steady improvements in acquisition time and resolution. As a result the sheer data throughput is continually increasing. Examples of techniques where the typical data output has moved from 1D to 3D in the past few decades are shown in Figure 1.

More data is always welcome. However, in many disciplines human digestion of these large amounts of data has now become the bottleneck. In many fields, for example those working at large scale synchrotron facilities where the duration of the experiment is limited, scientists require a means of quick data inspection and carrying out a fast preliminary analysis in order to take decisions on the course of the experiment. Many of the existing powerful and versatile visualization tools (Ahrens et al., 2005; Fedorov et al., 2012; *Mayavi*, n.d.; *VisIt*, n.d.) are not well suited for this purpose and cannot easily support the specific and often changing needs of a given discipline. The result is that each community ends up developing their own solutions to the problem of quick data visualization and inspection, e.g. (Lass et al., 2020; Stansbury & Lanzara, 2020). However, since these implementations are usually intertwined and entangled with the community-specific parts, such solutions are typically not transferrable across different disciplines or experimental methodologies.

We have developed PIT and the data-slicer package to account for these needs: offering tools for fast live visualization of data at a general scope that can easily be adjusted and fine tuned for more specific problems.

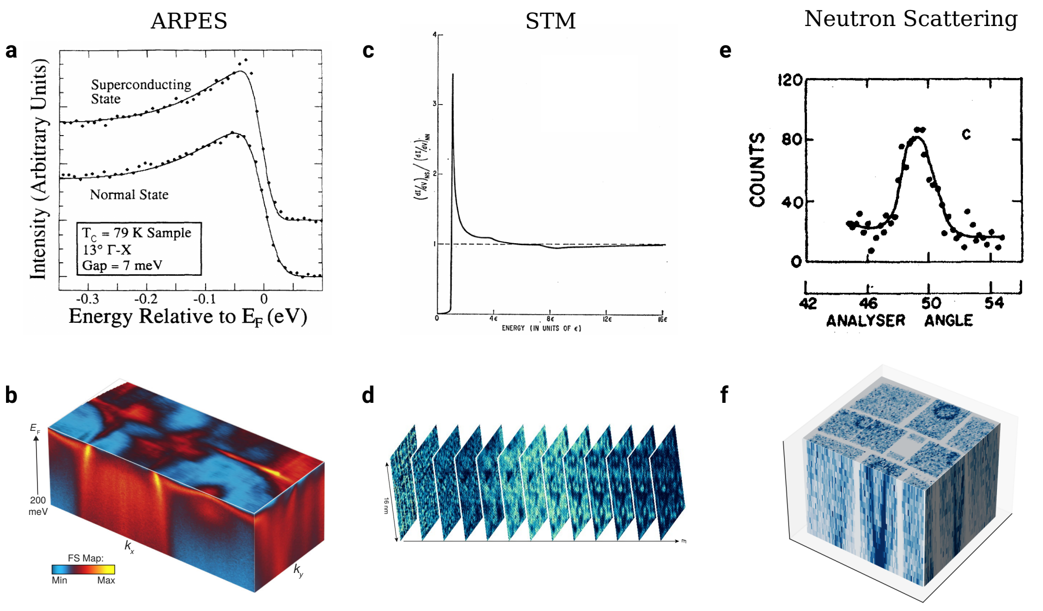

Figure 1: Evolution of data acquisition in the field of spectroscopy. (a,b) Angle resolved photoemission electron spectroscopy (ARPES) (Shai et al., 2013; Wells et al., 1992), (c,d) tunnelling spectroscopy (STS) (Giaever et al., 1962; Zhang et al., 2019), and (e,f) inelastic neutron scattering (INS) (Dalla Piazza et al., 2015; Wan et al., 2020; Woods et al., 1960) spectroscopy techniques all started with single spectrum collection (top row). Modern spectroscopic and scattering techniques, however, involve multidimensional data acquisition (bottom row).

Summary

data-slicer is a python package that contains several functions and classes which provide modular Qt (*Riverbank Computing*, 2020; *The Qt Company*, 2020) widgets, tools and utilities for the visualization of three-dimensional (3D) datasets. These building blocks can be combined freely to create new applications. Some of these building blocks are used within the package to form a graphical user interface (GUI) for 3D data visualization and manipulation: the Python Image Tool (PIT). The relation between different elements of the package and external software is schematically depicted in Figure 2.

PIT

PIT consists of a number of dynamic plot figures which allow browsing through 3D data by quickly selecting slices of variable thickness from the data cube and further cutting them up arbitrarily. Two core features of PIT should be explicitly mentioned. The first is the ability to create slices of the 3D data cube along arbitrary angles quickly. This is facilitated on the GUI side through a simple draggable line to select the slice direction. The superior speed of this operation is enabled by the use of optimized functions. The second feature worth mentioning is the inclusion of an ipython console which is aware of the loaded data as well as of all GUI elements. The console immediately enables users to run any analysis routine suitable to their respective needs. This includes running python commands in a workflow familiar to pylab or Jupyter (*The Jupyter Project*, 2020) notebook users but also loading or directly running scripts into or from the console, using ipython's line magic functions `%load` and `%run` respectively. Effectively, this design is central in empowering users to accomplish any task imaginable — as long as it is possible to achieve with python.

Plugins

It is clear that it can get complicated and tedious to run certain types of data processing or analysis from the ipython console, as described in the previous paragraph. For such cases, PIT provides an additional level of customizability and control through its plugin system. Plugins are regular python packages that can be loaded from within PIT and enhance it with new functionality. A plugin can interact with all elements in PIT via the same interfaces as can be done through the built-in ipython console. Creating a plugin therefore requires little programming skills and no further knowledge of the inner workings of PIT. In this manner, different communities of users can create and share their field-specific plugins which allow them to customize PIT to their needs.

As an example, we mention the ds-arpes-plugin (Kramer, 2020), which provides basic functionalities for loading of ARPES datasets and handles for typical analysis functions, customized and taylored to be used from within PIT.

Modularity and widgets

PIT is constructed in a modular fashion, constituting of different widgets that have been combined together to make a useful, ready to use tool. However, different applications may require slightly different functionalities, and the setup in PIT may not be optimal. The data-slicer package makes all the used widgets in PIT and some additional ones independently available to the user. These widgets can be arbitrarily combined to create customized applications in a relatively simple manner.

In summary, the data-slicer package solves the problem of offering the right scope – neither too specialized that it can only be used by a narrow community nor too bloated such that it becomes hard to do specific operations – by offering a variety of methods for users of varying backgrounds to get exactly the tools they need. On the first and most general level, PIT offers a ready-to-use GUI for quick 3D data visualization without any need of programmatic user input. Users can satisfy their more specific needs either through use of the console or by implementing a plugin, which can both be accomplished with little programming knowledge. On the last, most specific level users can use and arrange the building blocks contained in the package to create completely new applications or embed PIT or other parts of the data-slicer package into an existing application.

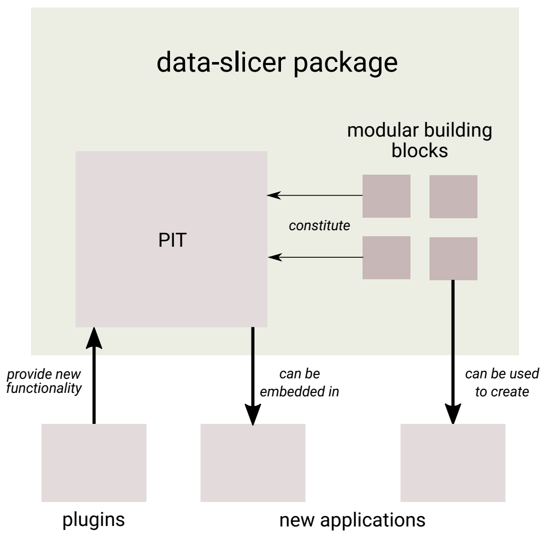

Figure 2: Schematic structural overview of the data-slicer package (represented by the large grey block). The python image tool (PIT) is made up of different modular building blocks. These blocks can be used to create or enhance new applications. PIT itself can be used directly, optionally augmented through plugins. Alternatively, it can be embedded in external applications.

Acknowledgements

We are thankful for fruitful discussions with and inputs from Titus Neupert, Claude Monney, Daniel Mazzone, Nicholas Plumb, Wojciech Pudełko, Niels Bech Christensen as well as for the testing efforts of Qisi Wang, Julia Küspert and Karin von Arx. We acknowledge the contribution of other open-source frameworks to this publication, namely matplotlib (Hunter, 2007), numpy (Harris et al., 2020) and pyqtgraph (*Pyqtgraph*, n.d.). Kevin Kramer and Johan Chang acknowledge support by the Swiss National Science Foundation.

References

Ahrens, J., Geveci, B., & Law, C. (2005). ParaView: An End-User Tool for Large Data Visualization. *Visualization Handbook*.

Dalla Piazza, B., Mourigal, M., Christensen, N. B., Nilsen, G. J., Tregenna-Piggott, P., Perring, T. G., Enderle, M., McMorrow, D. F., Ivanov, D. A., & Rønnow, H. M. (2015). Fractional excitations in the square-lattice quantum antiferromagnet. *Nature Physics*, *11*(1), 62–68. https://doi.org/10.1038/nphys3172

Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., Fillion-Robin, J.-C., Pujol, S., Bauer, C., Jennings, D., Fennessy, F., Sonka, M., Buatti, J., Aylward, S., Miller, J. V., Pieper, S., & Kikinis, R. (2012). 3D Slicer as an Image Computing Platform for the Quantitative Imaging Network. *Magn Reson Imaging*, *30*(9), 1323–1341. https://doi.org/10.1016/j.mri.2012.05.001

Giaever, I., Hart, H. R., & Megerle, K. (1962). Tunneling into Superconductors at Temperatures below 1 k. *Phys. Rev.*, *126*(3), 941–948. https://doi.org/10.1103/PhysRev.126.941

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., R'ıo, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kramer, K. (2020, December 2). *Ds-arpes-plugin documentation*. https://ds-arpes-plugin.readthedocs.io/en/latest/index.html

Lass, J., Jacobsen, H., Mazzone, D. G., & Lefmann, K. (2020). MJOLNIR: A software package for multiplexing neutron spectrometers. *SoftwareX*, *12*, 100600. https://doi.org/10.1016/j.softx.2020.100600

Mayavi. (n.d.). Retrieved February 25, 2021, from https://docs.enthought.com/mayavi/mayavi/

Pyqtgraph. (n.d.). Retrieved March 8, 2021, from http://www.pyqtgraph.org/

Riverbank Computing. (2020, November 27). https://riverbankcomputing.com/software/pyqt/intro

Shai, D. E., Adamo, C., Shen, D. W., Brooks, C. M., Harter, J. W., Monkman, E. J., Burganov, B., Schlom, D. G., & Shen, K. M. (2013). Quasiparticle Mass Enhancement and Temperature Dependence of the Electronic Structure of Ferromagnetic $\mathrm{SrRuO_3}$ Thin Films. *Phys. Rev. Lett.*, *110*(8), 087004. https://doi.org/10.1103/PhysRevLett.110.087004

Stansbury, C., & Lanzara, A. (2020). PyARPES: An analysis framework for multimodal angle-resolved photoemission spectroscopies. *SoftwareX*, *11*, 100472. https://doi.org/10.1016/j.softx.2020.100472

The jupyter project. (2020, November 9). https://ipython.org/

The qt company. (2020, November 27). https://www.qt.io

VisIt. (n.d.). Retrieved February 25, 2021, from https://wci.llnl.gov/simulation/computer-codes/visit

Wan, W., Christensen, N. B., Sandvik, A. W., Tregenna-Piggott, P., Nilsen, G. J., Mourigal, M., Perring, T. G., Frost, C. D., McMorrow, D. F., & Rønnow, H. M. (2020). Temperature dependence of the $(\pi, 0)$ anomaly in the excitation spectrum of the 2D quantum heisenberg antiferromagnet. *Journal of Physics: Condensed Matter*, *32*(37), 374007. https://doi.org/10.1088/1361-648x/ab757a

Wells, B. O., Shen, Z. X., Dessau, D. S., Spicer, W. E., Mitzi, D. B., Lombardo, L., Kapitulnik, A., & Arko, A. J. (1992). Evidence for k-dependent, in-plane anisotropy of the superconducting gap in $\mathrm{Bi_2Sr_2CaCu_2O_{8+\delta}}$. *Phys. Rev. B*, *46*(18), 11830–11834. https://doi.org/10.1103/PhysRevB.46.11830

Woods, A. D. B., Cochran, W., & Brockhouse, B. N. (1960). Lattice Dynamics of Alkali Halide Crystals. *Phys. Rev.*, *119*(3), 980–999. https://doi.org/10.1103/PhysRev.119.980

Zhang, Y., Mesaros, A., Fujita, K., Edkins, S. D., Hamidian, M. H., Ch'ng, K., Eisaki, H., Uchida, S., Davis, J. C. S., Khatami, E., & Kim, E.-A. (2019). Machine learning in electronic-quantum-matter imaging experiments. *Nature*, *570*(7762, 7762), 484–490. https://doi.org/10.1038/s41586-019-1319-8