Fine Structure and Dynamics of the Solar Atmosphere
Proceedings IAU Symposium No. 327, 2016
S. Vargas Domínguez, A.G. Kosovichev, P. Antolin & L. Harra, eds.

doi:10.1017/S1743921317003568

A Python-based interface to examine motions in time series of solar images

J. I. Campos-Rozo[1] **and S. Vargas Domínguez**[2]

[1]Observatorio Astronómico Nacional, Universidad Nacional de Colombia, Bogotá, Colombia
email: jicamposr@unal.edu.co

[2]Observatorio Astronómico Nacional, Universidad Nacional de Colombia, Bogotá, Colombia
email: svargasd@unal.edu.co

Abstract. Python is considered to be a mature programming language, besides of being widely accepted as an engaging option for scientific analysis in multiple areas, as will be presented in this work for the particular case of solar physics research. SunPy is an open-source library based on Python that has been recently developed to furnish software tools to solar data analysis and visualization. In this work we present a graphical user interface (GUI) based on Python and Qt to effectively compute proper motions for the analysis of time series of solar data. This user-friendly computing interface, that is intended to be incorporated to the Sunpy library, uses a local correlation tracking technique and some extra tools that allows the selection of different parameters to calculate, vizualize and analyze vector velocity fields of solar data, i.e. time series of solar filtergrams and magnetograms.

Keywords. GUI, Solar Physics, Python, Sunpy, LCT.

1. Introduction

Python is a powerful and easy-to-learn programming language. Nowadays, it is widely accepted that Python "is very efficient for reading high-level data structures and it is a very simple but effective approach to OOP (Object-oriented programming)". Python is an ideal language to quickly develop codes and applications. The Python interpreter and the large number of standard libraries are freely available from different sources or binaries for all Operative Systems (OS) on the Python's website. Python can be extended easily using C or C++ functions. Among the most representative aspects it is worth mentioning that it has several multipurpose tools, as Scikit-image (image processing package), scikit-learn (neural network), Mayavi (3-D visualization), Sympy (symbolic mathematics), Django (web development) and many others. Many of these attractive features have promoted Python as a very propitious language for science and data analysis in recent years, particulary used for efficient numerical computing (Langtangen, H. P. 2004).

The large number of Python packages constitute a well-developed computing environment. Focusing on scientific data computing, and particularly in astronomical analysis, the main scientific packages used are *Scipy*, *Numpy* , *Pandas*, *PyQt*.

1.1. *Sunpy*

Python has progressively increased its acceptance in the astronomy community, being researchers in solar physics largely responsable for this in the last five years. The Sunpy project is an effort to develop a package of open-source software for the data analysis and display the different kinds solar data. Sunpy is based on Python and uses Python

scientific packages founded on the philosophy of providing "the software tools necessary so that anyone can analyze solar data" (SunPy WebPage 2017). Based on the basic packages for handling arrangements of Numpy, numerical algorithms of Scipy (Jones *et al.* 2001) and displaying packages of Matplotlib (Hunter, J. D. 2007) the development of Astropy packages is possible (Astropy Collaboration *et al.* 2013), providing more specific functionalities. The Sunpy community is motivated, among others, by the need to provide modern and free tools as a replacement to the library SolarSoftware (SSW), based on the proprietary programming language IDL. Sunpy has developed three classes in consonance with three types of solar physics data: images, time series and spectra. These classes allow access to data and associated metadata and provide adequate comfort functions for the subsequent analysis and visualization (SunPy Community *et al.* 2015).

2. Graphical User Interface Design: A usable tool for LCT

In a previous work by Campos and Vargas Domínguez (2014), a preliminary version of a graphical user interface (GUI) was presented. Based on the former stage of the interface, we implemented a series of improvements and visualization tools. This work condensates a large number of tests, that allowed us to incorporate modifications to the original routines for an effective use of the GUI. The interface enables the user to easily compute the proper motions of structures in a time series of images, that are given as an input through a data cube, by using an algorithm based on a local correlation tracking technique. Although this study is based on solar data (i.e. filtergrams and magnetograms), the interface can be broadly used with any image data collection to measure proper motions.

2.1. *Basis of the Local Correlation Tracking (LCT) technique*

According to November and Simon (1988), the spatially localized cross correlation $C(\delta, \mathbf{x})$ is a 4-dimensional function: two dimensions are formed by the displacement vector δ and two dimensions are given by the central position of the window function $\mathbf{x}$, which maximizes the best cross correlation between two consecutive images. The cross correlation function $C(\delta, \mathbf{x})$ is defined in terms of the image intensity $J_t(\mathbf{x})$ and $J_{t+\tau}(\mathbf{x})$, which are two consecutive images of the data cube in consecutive time steps t and $t+\tau$:

$$C(\delta, \mathbf{x}) = \int J_t\left(\zeta - \frac{\delta}{2}\right) J_{t+\tau}\left(\zeta + \frac{\delta}{2}\right) W(x-\zeta) d\zeta \tag{2.1}$$

As mentioned in November and Simon (1988) the boundaries of the integral would include theoretically the full image. However, in practice it becomes limited by the effect of the apodizing window function $W(x)$. In the case of LCT algorithms this function is normally given by a Gaussian with a characteristic size that depends on the features which one wants to follow, and therefore to study their dynamics..

2.2. *How to use the GUI*

The Graphical User Interface (GUI) allows reading data cubes in FITS (Flexible Image Transport System) format or IDL formats. It is compulsory that the time series of images are properly pre-treated (de-rotation, alignment and filtering to correct from jittering) prior to the use of the GUI. SunPy provides functions to coalign a list of maps. The widget incorporates six different tools or workspaces (see Fig.1). The main workspace is the so-called *Flows Tools*, which is based on the LCT algorithm. This workspace is divided in two main sections. The first one (red box in the figure) is responsible for setting the most relevant parameters to calculate the proper motions of structures in the images (size of the correlation tracking window, pixel size of the images and cadence of

https://doi.org/10.1017/S1743921317003568 Published online by Cambridge University Press

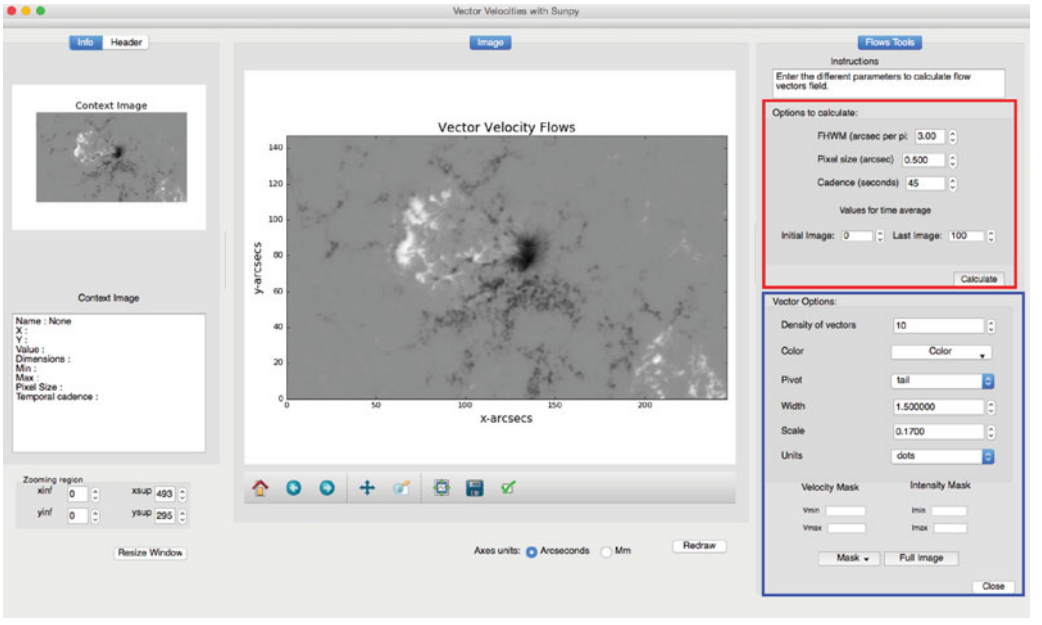

Figure 1. Python-based GUI for applying LCT analysis over a time series of images. Framed in the red are the more relevant parameters that can be adjusted to compute flow maps. The blue box frames the variables used for visualization of the flow fields.

the time series), and the second block (blue box) contains the visualization parameters needed for creating the flow fields (velocity vectors).

The widget includes additional tools for statistical analysis of the resulting flow fields, i.e. histograms of intensity as well as velocity distributions for rapid comparisons. Moreover, one can change images contrast and generate overlays to highlight peculiar regions, contour plots, vertical velocity maps (up- and downflows), masking appliances for easily exploring regions of interest, and the possibility to save all the resulting images. The GUI has been created using QtDesigner, a tool of Qt company, and converted and finished with PyQt (Riverbank Group 2016).

References

Astropy Collaboration, *et al.* 2013, *A&A*, 558, A33.
Campos Rozo, J. I. & Vargas Domínguez, S. 2014, *CEAB*, 38, 67-72.
Hunter, J. D. 2007, *AIP - Computing in Science & Engineering*, 9.
Jones, E., Oliphant, T., Peterson, P., *et al.* 2001, *SciPy: Open source scientific tools for Python. Online; accessed 2016-12-23*
Langtangen, H. P. 2004, *Python Scripting for Computational Science*, *Springer-Verlag Berlin Heidelberg.*
November, L. J. & Simon, G. W. 1988, *ApJ*, 333, 427-442.
Riverbank Group 2016, *https://www.riverbankcomputing.com/static/Docs/PyQt4/pyqtwhitepaper-us.pdf. Online; accessed 2016-12-22*
SunPy Community *et al.* 2015, *APJ - Computational Science and Discovery*, 8.
SunPy WebPage 2017, *http://sunpy.org/about/ Online; accessed 2017-03-13*

https://doi.org/10.1017/S1743921317003568 Published online by Cambridge University Press