pyLICORS Documentation

Release 0.1

Georg M. Goerg

December 04, 2012

CONTENTS

1 Pre-release of software and documentation **3**

2 Setup and Installation **5**
- 2.1 One-line installation . . . 5
- 2.2 Dependencies . . . 5

3 Documentation **7**
- 3.1 TODO and feature requests . . . 7

4 Indices and tables **9**

pyLICORS is a Python package estimates the predictive state space of a spatio-temporal process given spatio-temporal data. Typically this is a video (2D random field) observed for T time steps.

Note: References are arxiv.org/abs/1211.3760 and arxiv.org/abs/1206.2398 .

---

pyLICORS can handle

1. $(2+1)D$ space-time random fields (aka as videos = 2D images in time (1D))
   - sequences of images (`bmp`, `jpg`, or `png`)
   - videos (`avi`, `mpg`, or `mov`)
   - fMRI data formats (`.gz`, `.img`, `nii.gz`)
2. $(1+1)D$ space-time random fields (e.g. cellular automata. They are less common in practice, but good for simulations and visualization: think of an image where space is vertical and time goes from left to right).
   - comma separated `txt` or `dat` files, where each row is the time evolution of the system at space point r; column 1 is the entire system at time $t = 1$, column 2 the system at time $t = 2$, and so on.
   - images (`bmp`, `jpg`, or `png`)

Note: Although libraries used in pyLICORS are available for most platforms that run Python, it was developed on a Windows installation of Python. It has also been successfully tested on Ubuntu 11.10 (only difference that on Ubuntu built-in *cv* library was used for OpenCV; on Windows *pyopencv*).

These instruction may follow - sometimes unaware - a Windows style installation. In principle pyLICORS should not have any problems to run on Linux/MacOS.

If you do encounter problems, please let me know.

CHAPTER

ONE

PRE-RELEASE OF SOFTWARE AND DOCUMENTATION

This is a very first planning release of the software: the main features are not available yet, it is just an initial draft. Core functions will be added over the course of the next months.

This means that the functionality and the documentation will change in the future. In particular, function names, class names, functionality of classes, internal structures of the code, etc. **will** certainly **change** (so all follow the Python PEP8 naming conventions).

Warning: Users that want to use individual functions in the package to run specific parts of the analysis must be aware that their scripts **will break** when run with future releases of pyLICORS.

CHAPTER

TWO

SETUP AND INSTALLATION

pyLICORS was built and is maintained in Python 2.7. As far as I am aware it does not use any particular 2.7 feature of Python so pyLICORS should also work with other versions (at least 2.5+ should not pose problems; but: Python 2 or 3 and What's new in Python 3).

If you follow the instructions below and get **all** third-party extension/libraries (both Python and C/C++) to work properly, then chances are high it will work also on your Python version.

2.1 One-line installation

Open the command line and run

> pip install pyLICORS

or

> easy_install pyLICORS

2.2 Dependencies

Note: From experience during the devolopment the most difficult part was to get the Python wrappers for OpenCV to work properly.

OpenCV has a built in Python wrapper `cv`, which can be imported using `import cv`. However, the `pyopencv` wrapper using the Boost libraries which are 3-4 times faster on a Windows machine.

Although several third-party wrappers for OpenCV exist (e.g. ctypes-opencv or Swig Python Interface from OpenCV), I could only successfully use pyopencv in Windows.

If you can successfully call the OpenCV Kmeans++ from Python with another wrapper I would appreciate instructions (and a simple demo script) so I can include it another wrapper as an option in future relaeses.

2.2.1 Required

The most important non-Python software to install are:

- OpenCV library (version 2.2 [1]): collection of powerful and **fast** computer vision algorithms. **Absolutely necessary** to perform fast Kmeans++ (original paper on kmeans++).
- FFmpeg: cross-platform solution to record, convert and stream audio and video. This library is used to write the results of the analysis back to a video file.
- OpenGL: OpenGL for video writing/reading.
- Boost: free peer-reviewed portable C++ source libraries. Necessary for pyopencv.

Before installing pyLICORS the following Python packages **must** be installed and run successfully (installing in this order will avoid dependency problems between libraries):

- numpy and scipy: libraries for scientific computing.
- matplotlib: plotting library
- pyopengl: Python wrapper for OpenGL. Used for visvis.
- pyopencv: Python wrapper for the OpenCV library
- visvis: Video processing module for python. Necessary to read and access videos frame-by-frame. Requires pyopengl and OpenGL.
- PyTables: Data storage in HDF5 files with a convenient Python wrapper.
- psutil: interface for retrieving information on all running processes and system utilization (CPU, disk, memory, network).
- PIL: another plotting library

Note: If you know another library for Kmeans that can be used from Python and is *faster than OpenCV Kmeans*, please let me know.

2.2.2 Optional

Recommended packages for better visualization/individual checks of the analysis are:

- nibabel: read and write access to some common medical and neuroimaging file formats
- ViTables: GUI for browsing and editing files in PyTables' HDF5 and standard hdf5 formats.
- sklearn: powerful machine learning library with a huge selection of methods. In particular, if you can't get the OpenCV library to work, then you can use the Kmeans algorithm in sklearn. However, this will be much slower and can only be used for small data projects.
- munkres: implementation of the Munkres algorithm (also called the Hungarian algorithm or the Kuhn-Munkres algorithm), useful for solving the Assignment Problem.
- VirtualDub: An excellent program (at least on Windows) to view videos frame by frame (and do almost any other imaginable video processing).

Note: These packages are optional; they are **not necessary** for the core functionality of pyLICORS.

---

[1] I use version 2.2. As long as you successfully set up a Python wrapper with the OpenCV library the actual version should not matter (probably it should at least be 2.2+).

CHAPTER

THREE

DOCUMENTATION

3.1 TODO and feature requests

3.1.1 Data formats etc

1. load data from URLs and not necessarily from local folders
2. save the resuls in a different (pre-specified) folder
3. allow specification of filename plus path, and do internally a split

3.1.2 Data handling, number crunching

1. if new light cones are completely in old light cones, dont make a new hdf5 file but project old hdf5 entries down to the new light cones.
2. pyLICORS should figure out the frames per chunk number automatically from the data / computer specification

3.1.3 Coding

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*