The VorBin Package v3.1 (19/FEB/2020)

VorBin: Adaptive Voronoi Binning of Two Dimensional Data

This VorBin package is a Python implementation of the two-dimensional adaptive spatial binning method of Cappellari & Copin (2003). It uses Voronoi tessellations to bin data to a given minimum signal-to-noise ratio.

Attribution

If you use this software for your research, please cite Cappellari & Copin (2003) The BibTeX entry for the paper is:

```
@ARTICLE{Cappellari2003,
    author = {{Cappellari}, M. and {Copin}, Y.},
    title = "{Adaptive spatial binning of integral-field spectroscopic
        data using Voronoi tessellations}",
    journal = {MNRAS},
    eprint = {astro-ph/0302262},
    year = 2003,
    volume = 342,
    pages = {345-354},
    doi = {10.1046/j.1365-8711.2003.06541.x}
}
```

Installation

install with:

```
pip install vorbin
```

Without writing access to the global `site-packages` directory, use:

```
pip install --user vorbin
```

Usage Example

To learn how to use the package `VorBin` run `voronoi_2d_binning_example.py` in the `vorbin/examples` directory, within the main package installation folder inside `site-packages`, and read the detailed documentation in the docstring of the file `voronoi_2d_binning.py`, on PyPi or as PDF from https://purl.org/cappellari/software.

Perform the following simple steps to bin you own 2D data with minimal Python interaction:

1. Write your data vectors [X, Y, Signal, Noise] in the text file `voronoi_2d_binning_example.txt`, following the example provided;
2. Change the line `targetSN = 50.0` in the procedure `voronoi_2d_binning_example.py`, to specify the desired target S/N of your final bins;
3. Run the program `voronoi_2d_binning_example` and wait for the final plot to appear. The output is saved in the text file `voronoi_2d_binning_output.txt`. The last column BIN_NUM in the file is *all* that is needed to actually bin the data;
4. Read the documentation at the beginning of the file `voronoi_2d_binning.py` to fully understand the meaning of the various optional output parameters.

VorBin Purpose

Perform adaptive spatial binning of two-dimensional data to reach a chosen constant signal-to-noise ratio per bin. This program implements the algorithm described in section 5.1 of Cappellari & Copin (2003)

Calling Sequence

```
binNum, xBin, yBin, xBar, yBar, sn, nPixels, scale = \
    voronoi_2d_binning(x, y, signal, noise, targetSN,
                       cvt=True, pixelsize=None, plot=True,
                       quiet=True, sn_func=None, wvt=True)
```

Input Parameters

x: Vector containing the X coordinate of the pixels to bin. Arbitrary units can be used (e.g. arcsec or pixels). In what follows the term "pixel" refers to a given spatial element of the dataset (sometimes called "spaxel" in the IFS community): it can be an actual pixel of a CCD image, or a spectrum position along the slit of a long-slit spectrograph or in the field of view of an IFS (e.g. a lenslet or a fiber). It is assumed here that pixels are arranged in a regular grid, so that the pixel size is a well-defined quantity. The pixel grid, however, can contain holes (some pixels can be excluded from the binning) and can have an irregular boundary. See the above reference for an example and details.

y: Vector (same size as X) containing the Y coordinate of the pixels to bin.

signal: Vector (same size as X) containing the signal associated with each pixel, having coordinates (X,Y). If the "pixels" are actually the apertures of an integral-field spectrograph, then the signal can be defined as the average flux in the spectral range under study, for each aperture. If pixels are the actual pixels of the CCD in a galaxy image, the signal will be simply the counts in each pixel.

noise: Vector (same size as X) containing the corresponding noise (1 sigma error) associated with each pixel.

targetsn: The desired signal-to-noise ratio in the final 2D-binned data. E.g. a S/N~50 per pixel may be a reasonable value to extract stellar kinematics information from galaxy spectra.

Optional Keywords

cvt: Set this keyword to skip the Centroidal Voronoi Tessellation (CVT) step (vii) of the algorithm in Section 5.1 of Cappellari & Copin (2003). This may be useful if the noise is strongly non-Poissonian, the pixels are not optimally weighted, and the CVT step appears to introduce significant gradients in the S/N. A similar alternative consists of using the /WVT keyword below. PLOT: Set this keyword to produce a plot of the two-dimensional bins and of the corresponding S/N at the end of the computation.

pixsize: Optional pixel scale of the input data. This can be the size of a pixel of an image or the size of a spaxel or lenslet in an integral-field spectrograph.

The value is computed automatically by the program, but this can take a long time when (X, Y) have many elements. In those cases, the PIXSIZE keyword should be given.

sn_func: Generic function to calculate the S/N of a bin with spaxels "index" with the form: "sn = func(index, signal, noise)". If this keyword is not set, or is set to None, the program uses the _sn_func(), included in the program file, but another function can be adopted if needed. See the documentation of _sn_func() for more details.

quiet: by default, the program shows the progress while accreting pixels and then while iterating the CVT. Set this keyword to avoid printing progress results.

wvt: When this keyword is set, the routine bin2d_cvt_equal_mass is modified as proposed by Diehl & Statler (2006, MNRAS, 368, 497). In this case the final step of the algorithm, after the bin-accretion stage, is not a modified Centroidal Voronoi Tessellation, but it uses a Weighted Voronoi Tessellation. This may be useful if the noise is strongly non-Poissonian, the pixels are not optimally weighted, and the CVT step appears to introduce significant gradients in the S/N. A similar alternative consists of using the /NO_CVT keyword above. If you use the /WVT keyword you should also include a reference to "the WVT modification proposed by Diehl & Statler (2006)."

Output Parameters

binnumber: Vector (same size as X) containing the bin number assigned to each input pixel. The index goes from zero to Nbins-1. IMPORTANT: THIS VECTOR ALONE IS ENOUGH TO MAKE *ANY* SUBSEQUENT COMPUTATION ON THE BINNED DATA. EVERYTHING ELSE IS OPTIONAL!

xbin: Vector (size Nbins) of the X coordinates of the bin generators. These generators uniquely define the Voronoi tessellation. Note: USAGE OF THIS VECTOR IS DEPRECATED AS IT CAN CAUSE CONFUSION

ybin: Vector (size Nbins) of Y coordinates of the bin generators. Note: USAGE OF THIS VECTOR IS DEPRECATED AS IT CAN CAUSE CONFUSION

xbar: Vector (size Nbins) of X coordinates of the bins luminosity weighted centroids. Useful for plotting interpolated data.

ybar: Vector (size Nbins) of Y coordinates of the bins luminosity weighted centroids.

sn: Vector (size Nbins) with the final SN of each bin.

npixels: Vector (size Nbins) with the number of pixels of each bin.

scale: Vector (size Nbins) with the scale length of the Weighted Voronoi Tessellation, when the /WVT keyword is set. In that case SCALE is *needed* together with the coordinates XBIN and YBIN of the generators, to compute the tessellation (but one can also simply use the BINNUMBER vector).

When some pixels have no signal

Binning should not be used blindly when some pixels contain significant noise but virtually no signal. This situation may happen e.g. when extracting the gas kinematics from observed galaxy spectra. One way of using voronoi_2d_binning consists of first selecting the pixels with S/N above a minimum threshold and then binning each set of connected pixels *separately*. Alternatively one may optimally weight the pixels before binning. For details, see Sec. 2.1 of Cappellari & Copin (2003).

Binning X-ray data

For X-ray data, or other data coming from photon-counting devices the noise is generally accurately Poissonian. In the Poissonian case, the S/N in a bin can never decrease by adding a pixel (see Sec.2.1 of Cappellari & Copin 2003), and it is preferable to bin the data *without* first removing the observed pixels with no signal.

Binning very big images

Computation time in voronoi_2d_binning scales nearly as npixels^1.5, so it may become a problem for large images (e.g. at the time of writing npixels > 1000x1000). Let's assume that we really need to bin the image as a whole and that the S/N in a significant number of pixels is well above our target S/N. As for many other computational problems, a way to radically decrease the computation time consists of proceeding in a hierarchical manner. Suppose for example we have a 4000x4000 pixels image, we can do the following:

1. Rebin the image regularly (e.g. in groups of 8x8 pixels) to a manageable size of 500x500 pixels;
2. Apply the standard Voronoi 2D-binning procedure to the 500x500 image;
3. Transform all unbinned pixels (which already have enough S/N) of the 500x500 Voronoi 2D-binned image back into their original individual full-resolution pixels;
4. Now apply Voronoi 2D-binning only to the connected regions of full-resolution pixels;
5. Merge the set of lower resolution bins with the higher resolution ones.

Changelog

V3.1.4: MC, Oxford, 19 February 2020

- Formatted documentation as docstring.

V3.1.3: MC, Oxford, 27 September 2018

- Fixed clock DeprecationWarning in Python 3.7.

V3.1.2: MC, Oxford, 10 May 2018

- Dropped legacy Python 2.7 support.

V3.1.1: MC, Oxford, 15 September 2017

- Stops if unbinned pixels do not have enough S/N.
- Removed weighted centroid function.

V3.1.0: MC, Oxford, 17 July 2017

- Use cKDTree for much faster un-weighted Voronoi Tessellation.
- Removed loop over bins from Lloyd's algorithm with CVT.

V3.0.9: MC, Oxford, 10 July 2017

- Do not iterate down to diff==0 in _cvt_equal_mass().
- Request pixelsize when dataset is large. Thanks to Davor Krajnovic (Potsdam) for the feedback.
- Make quiet really quiet.
- Fixed some instances where sn_func() was not being used (only relevant when passing the sn_func keyword).

V3.0.8: MC, Oxford, 15 February 2017

- New voronoi_tessellation() function.

V3.0.7: MC, Oxford, 23 January 2017

- Print execution time.

V3.0.6: MC, Oxford, 14 June 2016

- Use interpolation='nearest' to avoid crash on MacOS. Thanks to Kyle Westfall (Portsmouth) for reporting the problem.
- Allow for zero noise.

V3.0.5: MC, Oxford, 18 April 2016

- Fixed deprecation warning in Numpy 1.11.

V3.0.4: MC, Oxford, 12 April 2016

- Included keyword "sn_func" to pass a function which calculates the S/N of a bin, rather than editing _sn_func().
- Included test to prevent the addition of a pixel from ever decreasing the S/N during the accretion stage.

V3.0.3: MC, Oxford, 31 March 2016

- Use for loop to calculate Voronoi tessellation of large arrays to reduce memory usage. Thanks to Peter Weilbacher (Potsdam) for reporting the problem and providing the solution.

V3.0.2: MC, Oxford, 2 October 2014

- Avoid potential runtime warning while plotting.

V3.0.1: MC, Oxford, 25 May 2014

- Support both Python 2.7 and Python 3.

V3.0.0: MC, London, 19 March 2014

- Translated from IDL into Python and tested against the original.

V2.6.0: MC, London, 19 March 2014

- Included new SN_FUNCTION to illustrate the fact that the user can define his own function to estimate the S/N of a bin if needed.

V2.5.8: MC, La Palma, 15 May 2012

- Update Voronoi tessellation at the exit of bin2d_cvt_equal_mass. This is only done when using /WVT, as DIFF may not be zero at the last iteration.

V2.5.7: MC, Oxford, 24 March 2012

- Included safety termination criterion of Lloyd algorithm to prevent loops using /WVT.

V2.5.6: MC, Oxford, 11 November 2011

- Use IDL intrinsic function DISTANCE_MEASURE for automatic pixelSize, when PIXSIZE keyword is not given.

V2.5.5: MC, Oxford, 28 April 2010

- Added PIXSIZE keyword.

V2.5.4: MC, Oxford, 30 November 2009

- Improved color shuffling for final plot.

V2.5.3: MC, Oxford, 3 December 2007

- Fixed program stop, introduced in V2.5.0, with /NO_CVT keyword.

V2.5.2: MC, Oxford, 28 March 2007

- Print number of unbinned pixels.

V2.5.1: MC, Oxford, 3 November 2006

- Updated documentation.

V2.5.0: MC, Leiden, 9 March 2006

- Added two new lines of code and the corresponding /WVT keyword to implement the nice modification to the algorithm proposed by Diehl & Statler (2006).

V2.4.8: MC, Leiden, 23 December 2005

- Use geometric centroid of a bin during the bin-accretion stage, to allow the routine to deal with negative signal (e.g. in background-subtracted X-ray images). Thanks to Steven Diehl for pointing out the usefulness of dealing with negative signal.

V2.4.7: MC, Leiden, 27 September 2005

- Verify that SIGNAL and NOISE are non negative vectors.

V2.4.6: MC, Leiden, 27 August 2005

- Added /NO_CVT keyword to optionally skip the CVT step of the algorithm.

V2.4.5: MC, Leiden, 3 December 2004

- Added BIN2D prefix to internal routines to avoid possible naming conflicts.

V2.4.4: MC, Leiden, 30 November 2004

- Prevent division by zero for pixels with signal=0 and noise=sqrt(signal)=0, as can happen from X-ray data.

V2.4.3: MC, Leiden, 29 November 2004

- Corrected bug introduced in version 2.3.1. It went undetected for a long time because it could only happen in special conditions. Now we recompute the index of the good bins after computing all centroids of the reassigned bins in reassign_bad_bins. Many thanks to Simona Ghizzardi for her clear analysis of the problem and the solution.

V2.4.2: MC, Leiden, 4 August 2004

- Use LONARR instead of INTARR to define the CLASS vector, to be able to deal with big images. Thanks to Tom Statler.

V2.4.1: MC, Leiden, 14 December 2003

- Added /QUIET keyword and verbose output during the computation. After suggestion by Richard McDermid.

V2.4.0: MC, Leiden, 10 December 2003

- Addedd basic error checking of input S/N.
- Reintroduced the treatment for zero-size bins in CVT, which was deleted in V2.2. Thanks to Robert Sharp and Kambiz Fathi for reporting problems.

V2.3.1: MC, Leiden, 13 April 2003

- Do *not* assume the first bin is made of one single pixel.
- Added computation of S/N scatter and plotting of 1-pixel bins.

V2.3.0: MC, Leiden, 9 April 2003

- Unified the three tests to stop the accretion of one bin. This can improve some bins at the border.

V2.2.0: MC, Leiden, 11 March 2003

- Added computation of useful bin quantities in output. Deleted some safety checks for zero size bins in CVT. Minor polishing of the code.

V2.1.0: MC, Vicenza, 13 February 2003

- First released version. Written documentation.

V2.0.0: MC, Leiden, 11 September 2001

- Major revisions. Stable version.

V1.0.0: Michele Cappellari, Leiden, June 2001

- First working implementation.