Weighted Voronoi Tessellations

Carter Rhea

October 15, 2019

1 Weighted Voronoi Tessellations

1.1 A Brief Introduction

Before delving into Weighted Voronoi Tessellations, I would first like to simply describe why we care about them (other than the fact that they are pretty cool). In a few words: WVT allows for a binning scheme that is unbiased when compared with Voronoi Tessellations, Centroidal Voronoi Tesselations, or other binning methods. Once we have the WVT bins in place, we can use that information to generate temperature maps for X-ray data (That's what I am interested in).

See [1], [2], and [3] for more information.

1.2 Signal-to-Noise

For our particular application in X-ray astronomy, we are going to be interested in a signal-to-noise ratio dominated by count rates. Count Rates are a relatively easy quantity for us to compute. The following equation are for pixel number k.

$$Signal_k = Flux_k = \frac{C_k}{E_k \tau} - B_k \tag{1}$$

$$Noise_k = \sigma_k^2 = \frac{C_k}{E_k^2 \tau^2} + \sigma_{B_k}^2 \tag{2}$$

Parameter	Definition
C_k	Raw Counts
E_k	Effective Area
τ	Exposure Time
B_k	Background Flux
σ_{B_k}	Uncertainty in Background Flux

$$B_k = \frac{\bar{C}_B}{E_k * \tau} \tag{3}$$

with $\bar{C}_B$ being the average background counts.

σ_{B_k} is just the $1 - sigma$ error value for B_k.

CURRENTLY BACKGROUND IS NOT TAKEN INTO ACCOUNT. However, this can be rectified by simply pass a background subtracted image[1].

This calculation and data-gathering occurs in the READ_IN subroutine.

That sums up signal-to-noise calculations in the current version of the code.

1.3 Algorithm Run-Down

This section will contain the pseudo-algorithm of my python code.

Before we are able to begin our algorithm (WVT) in earnest, we must have a decent initial guess!

Enter **Bin Accretion**....

[1] http://cxc.harvard.edu/ciao/ahelp/blanksky_image.html

1.3.1 Bin Accretion

Data: Pixels
Result: Initial Guess for WVT
initialization;
while *Pixels left to be assigned* **do**
  while *Criteria not all met* **do**
    Calculate pixel closest to bin centroid;
    Calculate Adjacency;
    Calculate Roundness;
    Calculate Potential Signal-to-Noise;
    if *Criteria all met* **then**
      Add pixel to Bin;
    else
      Start new bin;
    end
    if *Signal-to-Noise too low* **then**
      Mark pixels unassigned;
    end
  end
end
Reassign all unassigned pixels to nearest bin;

Algorithm 1: Bin Accretion Algorithm

With that out of the way we can discuss how I actually calculated the three different criteria: Adjacency, Roundness, Potential Signal-to-Noise.

- Adjaceny: This one is simple: just check if the pixels are neighbors!
- Roundness: Ok so now we finally need to calculate some stuff... More explicity R_{equiv} and R_{max}. R_{max} is the maximum distance from the centroid for ALL pixels in bin. R_{equiv} is a wonderfully horrendous quantity which is "the radius of a disk around the bin". Lets do a quick little derivation...

$$Area_{circle} = \pi r_{circle}^2 == n\pi \frac{d_{pix}}{2} r = Area_{bin} \tag{4}$$

  Solving for r_{circle} we get,

$$r_{circle} = \sqrt{\frac{n}{\pi}}\, d \tag{5}$$

  And now we can calculate the roundness parameter which describes the compactness of the bin:

$$Roundness = \frac{R_{max}}{R_{equiv}} - 1 \tag{6}$$

  We generally shoot for a value of $Roundness = 0.3$.

- Potential Signal-to-Noise: We add the pixel to the bin and then recompute the Signal-to-Noise. This needs to be less than the target Signal-to-Noise.

And now to the main event (which is not very much coding wise compared to bin accretion).

1.3.2 WVT

Data: Bins
Result: WVT Data
initialization $--> $ read in Bin Accretion Data;
while *Bins not converged* **do**
    for *bin in Bins* **do**
        Recalculate Signal-to-Noise;
        Calculate Area;
        Calculate Centroid;
        Calculate Scale Length;
    end
    Reassign Pixel to Closest Bin;
end

Algorithm 2: Weighted Voronoi Tessellation Algorithm

And again we need to define two properties: Area and Scale Length.

- Area: Area of bin. This is basically just the area of the pixels in the bin...

$$A_{bin} = (Pixel\_Length)^2 * N_{pixels} \tag{7}$$

- Scale Length: This scale length is the KEY to the WEIGHTED part of WVT.

$$\sqrt{\frac{A_{bin}}{\pi}\frac{S/N}{(S/N)_{target}}} \tag{8}$$

And really thats it!

2 Preprocessing of Data and Selection of Inputs

I finally decided to put this section as the second chapter since the main goal of this document is to outline the WVT algorithm. However, I also wanted to include this information for anyone interested in X-ray data reduction from *CHANDRA*.

More importantly, this section follows the steps necessary to create all the inputs for the WVT algorithm.

2.1 Download Data

Before we can begin to preprocess the data, we obviously need to have the data. Thankfully, this is incredibly simple! After downloading the data from the chandra website or using

$$download\_chandra\_obsid\ \#OBSID$$

There exists a considerable amount of documentation on each file downloaded at

$$http://cxc.harvard.edu/ciao/threads/intro\_data/$$

2.2 Reduce Data

We reduced the data with the following command:

$$chandra\_repro$$

For details on what the reprocessing does visit the following website:

$$http://cxc.harvard.edu/ciao/ahelp/chandra\_repro.html$$

2.3 Region of Interest

Once we have the reduction completed, we can move onto dealing with the data we are interested in...

MUST BE RECTANGULAR REGION FILES

Create your region of interest as such...

- opened ds9 and picked the region of interest: "ds9 ...evt2.fits &" and RofI is called "simple.reg"
- Also created background image "simple_bkg.reg"

We must save the *simple.reg* as standard, `ftk5`, units **AND** in `IMAGE` units. Save the `IMAGE` coordinate region file as *simple_imcoord.reg*.

2.4 Create Fits Files and Exposure Map

At this point, we have all inputs necessary to run another python file: `Precursor.py` – found in **AstronomyTools/GeneralUse**. Running this tool will create both fits files for the region of interest and the exposure map. If you opt to not use `Precursor.py`, run the following commands...

Now we must create an image-fits file by running the following commands:

```
dmcopy "acisf#####_repro_evt2.fits[sky=region(simple.reg)]" simple.fits
dmcopy "simple.fits[events][bin x=::.1,y=::.1][IMAGE]" simple_img.fits
dmcopy "acisf#####_repro_evt2.fits[sky=region(simple_bkg.reg)]" simple_bkg.fits
dmcopy "simple_bkg.fits[events][bin x=::.1,y=::.1][IMAGE]" simple_bkg_img.fits
```

And now we create the exposure map with the following command....

```
fluximage "simple.fits" outroot=flux bands=broad binsize=1 units=area
    clobber=yes cleanup=no
```

Now we move on to the other inputs....

- StN_Target - This is a user chosen value for the signal-to-noise ratio. This is highly dependent on how many counts you have in your specific data set, but a good goal is generally around 50. You may have to experiment a little with this value!!!!
- pixel_size - This is the pixel radius in degrees for a given CCD. If you are using a Chandra ACIS I observation (which for the team I am a part of is quite likely), simply retain the default value of 0.5 (yes yes yes I know... its 0.492 but 0.5 works for this)...
- ToL - Tolerance level for convergence. We calculate the convergence criteria based on how cluster the final bins signal-to-noise ratio is to the target ratio. Usually a value of at least 0.01 works well. If it takes more than 20 steps to converge, the program will simply terminate with the current StN values.
- roundness_crit - This parameter determines how "round" the initial bins are from the bin accretion portion of the algorithm. I only included this for those who really care. For the rest of us, I would advise using a value of 0.3.

3 Merged Observations

If the observations are merged we have to simply use the file created when `merge_obs` is run[2]; We must also have a region file (i.e. *region.reg*). Finally we can create our image for the region with the following command:

$$dmcopy"evt.fits[EVENTS][sky = region(region.reg)]"region.imgopt = image$$

Then you can feed *region.img* into the WVT input file.

3.1 NGC4636

The remainder of this section is an example using **NGC4636**. All the relavent files necessary to run this example can be found in **WVT/Verifications/NGC4636**.

3.1.1 Data Cleaning

We downloaded the *Chandra* data for **NGC4636** which correspond to OBSIDS 323 and 324 using `download_chandra_obsid 323,324`. We then reprocessed the data using `chandra_repro`; since we are not interested in the actual science we did not subtract background flares. We then chose a region which well captured the cluster called *center.reg*. Finally we created the *center.img* file with the procedure described above.

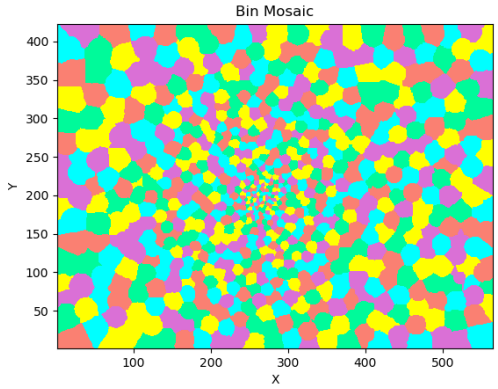
3.1.2 Input file

```
#Sample input file for WVT.py
#All lines without equals sign are ignored...
image_fits = center.img
exposure_map = none
StN_Target = 20
pixel_radius = 0.5
ToL = 1e-4
roundness_crit = 0.3
home_dir = /home/carterrhea/Desktop/WVT_test/Merged_unbinned
image_dir = /home/carterrhea/Desktop/WVT_test/Merged_unbinned
output_dir = /home/carterrhea/Desktop/WVT_test/Merged_unbinned
```

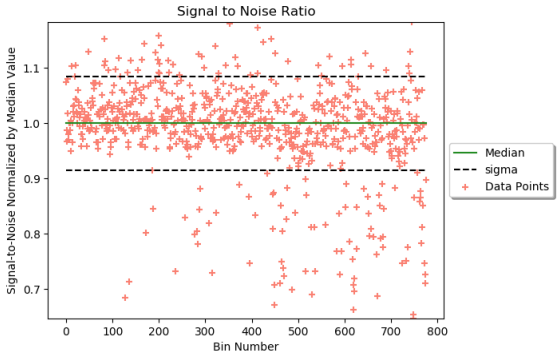
3.1.3 Results

Here we demonstrate the resulting image and statistics after the bin accretion step and after the final step.

[2]Be sure to set `binsize=1`

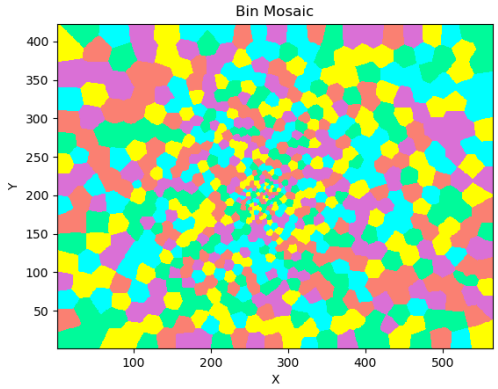


(a) Bin Mosaic

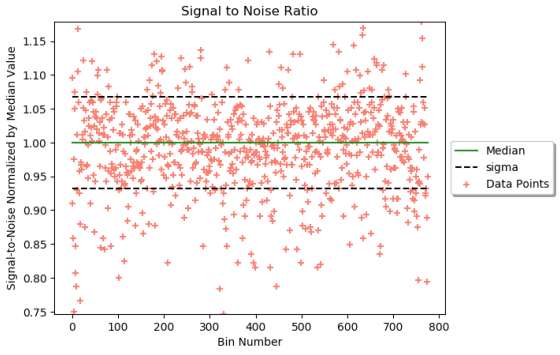


(b) Signal-to-Noise Scatter

Figure 1: Bin Accretion Step



(a) Bin Mosaic



(b) Signal-to-Noise Scatter

Figure 2: Final Step

Bibliographies

References

[1] Yannick Copin. Adaptive spatial binning of integral-field spectroscopic data using Voronoi tessellations. 10(January):1–10, 2018.

[2] Steven Diehl and Thomas S Statler. Adaptive Binning of X-ray data with Weighted Voronoi Tesselations. 14(February):1–14, 2008.

[3] J S Sanders and A C Fabian. Adaptive binning of X-ray galaxy cluster images. 000(May), 2018.