All-Sky Wavelet-Based Point Source Detection

Created	@April 30, 2023 10:54 AM
Status	Open
Updated	@May 1, 2023 11:24 AM

Explanation of skimage peak_local_max

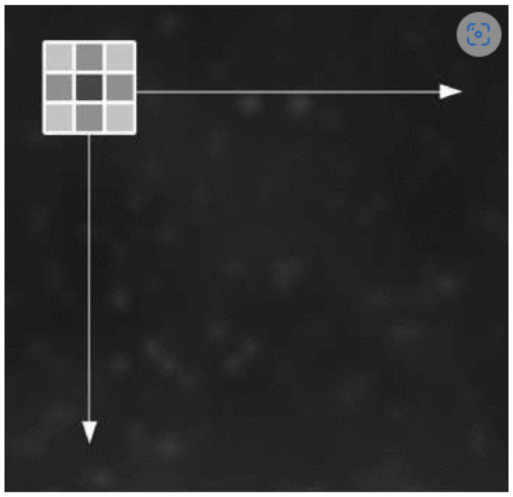
To detect peaks in our wavelet maps, we use the *skimage.peak_local_max* function. We treat our grid of wavelet coefficients as an image. The *skimage.peak_local_max* has two major steps:

1. Apply a maximum filter to the data
2. Determine peaks from the maximum filter

A *maximum filter* works as follows. It assigns to each pixel position the intensity of the brightest neighboring pixel. Representing the intensity of the pixels with a matrix I_{ij} and the neighbors of a pixel with a general index N, the maximum filter then produces

$$I_{ij} \rightarrow \max_N I_{i_N j_N}$$

How the "neighbors" are defined is a matter of definition. For the case of the nearest-neighbors, the act of applying the maximum filter across the whole map would scan through the image like the schematic below:



The *skimage.peak_local_max* function defines the "neighbors" of a center pixel as the pixels that are some distance (in units of pixel number) away from the center pixel. This is specified by the *min_distance* argument.

With the maximum filter applied, *peak_local_max* identifies peaks of an image with pixels whose intensity in the original map are equal to their intensities in the maximum-filter map. I.e., the peak positions (i_p, j_p) are defined by:

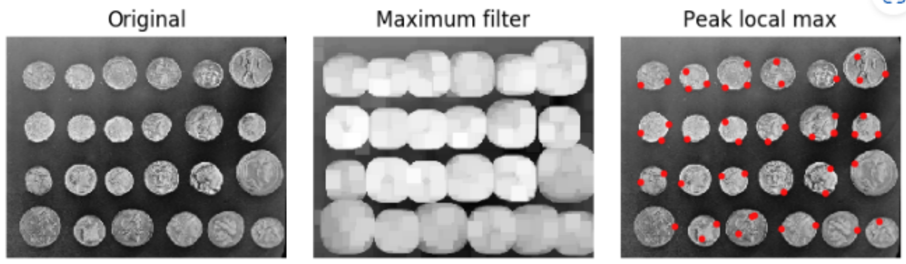
$$I_{i_p j_p} = \max_{N(p)} I_{i_{N(p)} j_{N(p)}}$$

In addition to tuning the set of neighbors in the maximum filter with the *min_distance* argument, one can tune other properties of the peak finder:

- Require a minimum intensity to peaks - *threshold_abs*
- Require a minimum intensity of peaks relative to maximum intensity of image - *threshold_rel*
- Exclude peaks that are within a set number of pixels from the border of the image - *exclude_border*
- Set a maximum to the number of detected peaks, prioritizing the brightest peaks - *num_peaks*
- Set norm used to define distance - *p_norm*
- Other options
  - scikit-image/peak.py at main · scikit-image/scikit-image (github.com)

In practice, we have been using the default parameters with *min_distance* $\sim 0.4^\circ$ in pixel units.

To see how the algorithm works, we reiterate the example given in the documentation:



As you can see, the maximum filter dilates the image of coins. The maximum filter involves comparing neighbors that are 20 pixels away from the center pixel. Moreover, the *peak_local_max* will have its *min_distance* = 20 pixels. This ensures that each blotch in the dilated image correspond to only one peak.

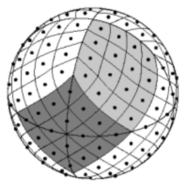
Continuous Wavelet Transform on the Whole Sphere

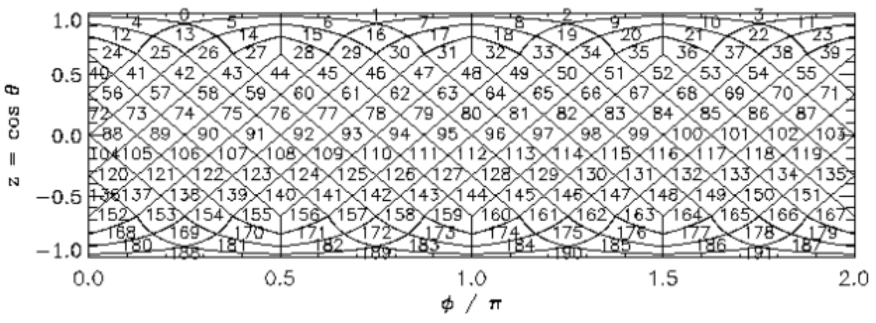
Two options:

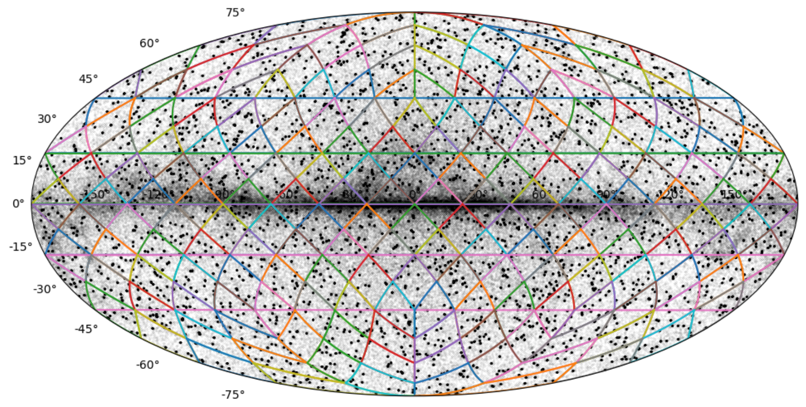
1. Invoke spherical wavelets
2. Apply standard 2D wavelets in small patches of the sky

We consider the second approach, as it allows us to implement our current code for flat wavelets and gives us greater freedom to parallelize the code and optimize our wavelet scales with respect to the background at each pixel.

Suppose we have a sample of photon counts across the whole sky, ignoring energy. To use the flat CWT for point source detection, the main idea is to decompose the sky into small patches, so that the difference between the difference between the Euclidean and angular diameter distance between two points in any given patch is negligible. As in Vielva 03, we define our patches using the HEALPIX scheme. Specifically, we partition the sky into 192, $\sim 10^\circ$ *father pixels* by setting NSIDE = 4.

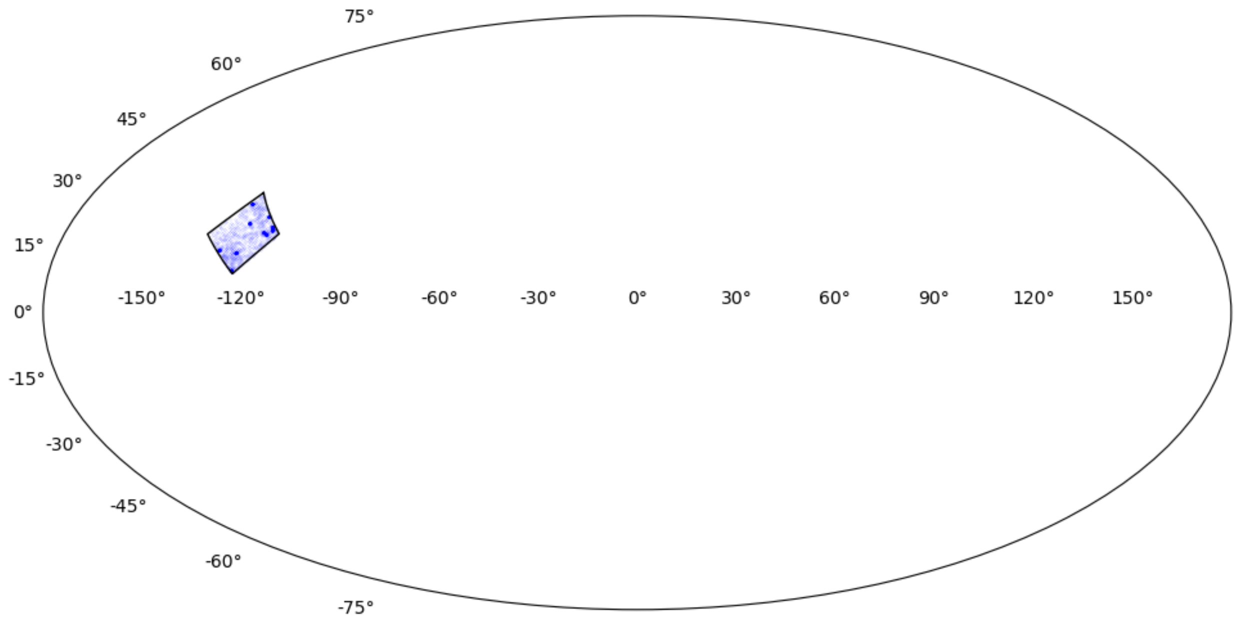






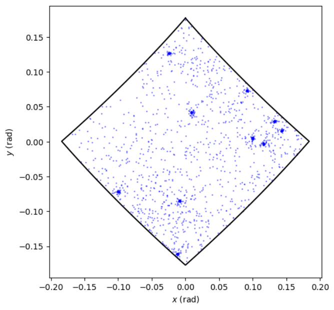
(Top Left) HEALPIX decomposition of sphere for NSIDE = 4 ; (Top Right) Labelling of pixels for NSIDE = 4 ; (Bottom) Background + Injected Point sources with HEALPIX father pixels overlaid

Then, we can group the photon counts based on which father pixel they occupy. We consider the 58th father pixel: npix = 58.



Father pixel corresponding to npix = 58 and the photon counts lying within it.

We perform the flat CWT on the father pixel in the following way. First, we project all the points on the father pixel to the plane tangent to the "center" of the father pixel. We take the central longitude (latitude) to be the midpoint between the max/min of the longitudes (latitudes). This transforms $(\theta, \phi) \to (x, y)$, where the origin corresponds to the center of the pixel, x denotes longitude, and y denotes latitude. We project the above father pixel below.



Projection of npix = 58 father pixel to the tangent plane defined by center of the pixel.

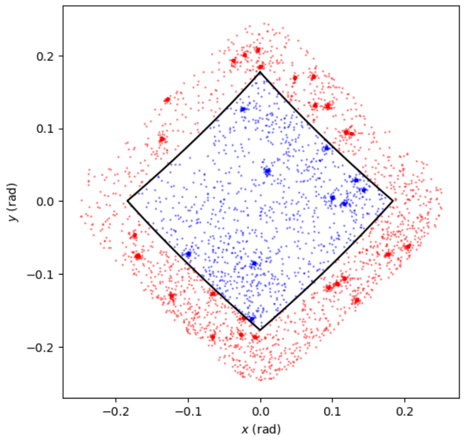
Once we draw a grid on the projected father pixel, we can use the same flat CWT technique as before to identify point sources. In summary, the algorithm has three steps:

1. Calculate wavelet coefficients at each grid point
2. Threshold map of wavelet coefficients
3. Identify peaks in the map using skimage.peak_local_max
4. Associating these peaks with point sources, we can obtain their positions in the sky by projecting the peak from the tangent plane back into the sphere

So far, we cannot apply this algorithm for each projected father pixel and expect valid results. Regarding the CWT calculation, the CWT at a point inside a given father pixel depends on points outside the father pixel. Regarding the peak identification algorithm, the peak finder behaves poorly at the boundaries of a given image.

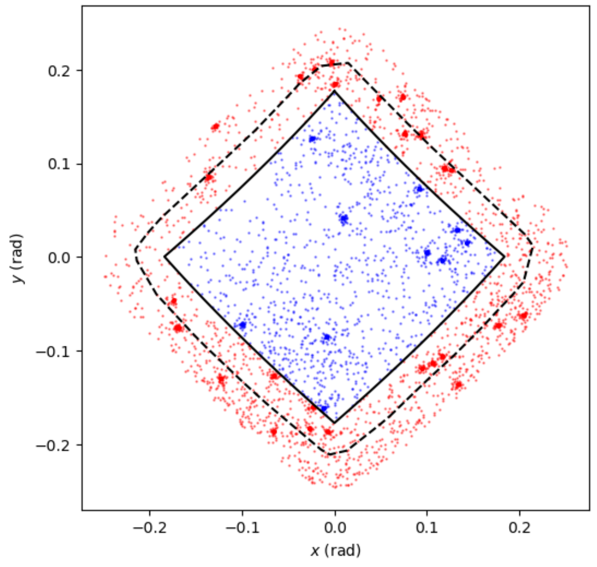
Since these issues are related to the presence of a boundary, we resolve it in the following way. Using the available data, we can find all points that are a given angular distance away from the father pixel. We take this distance to be $\sim 4^{\circ}$, as the base Mexican Hat Wavelet is very small relative to its local min/max for $r > 5$. Since we found wavelets of scales $\gtrsim 0.8^{\circ}$ do not perform well at discriminating sources from background, this is a reasonable starting point. A more careful analysis should show that the list of candidate point sources does not strongly depend on increases to this angular scale.

After projecting these extra points to the tangent plane, they define a band of data surrounding the father pixel.

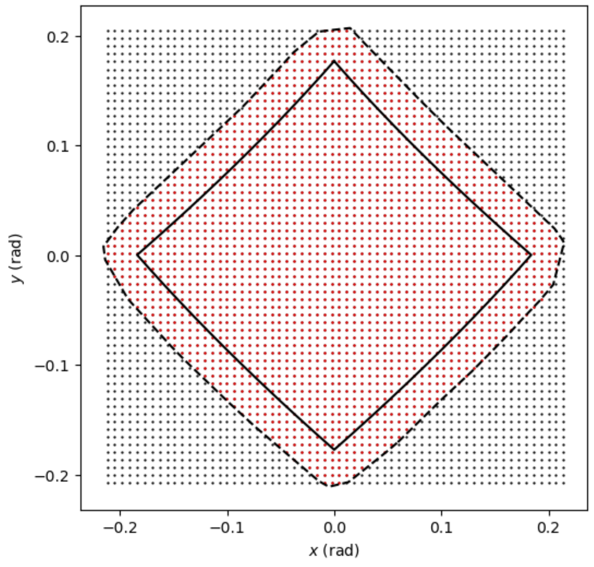


npix = 58 father pixel with additional band of points for accurate CWTs at the boundaries of the father pixel.

By including these extra points in our CWT calculations, we calculate more accurate estimates of the CWT near the edges of a father pixel. However, for the peak identification algorithm, the issue is not including data that is outside the boundary. It is merely not having any grid points outside the boundary. The question now is how we should define grid points that are outside the father pixel. To do this, we actually make use of the surrounding points. If the data is reasonably dense outside the father pixel, the boundary of the data will define a curve that encloses the father pixel. These curves are well-behaved for the background maps that we have used so far. We can use this boundary to define a larger set of grid points for the CWT calculation, which are inside and outside of the father pixel. In practice, we consider the boundary of points that are $\sim 2^\circ$ away from the father pixel since the only role these grid points serve is for the peak identifier to perform well on the actual boundaries of the father pixel.

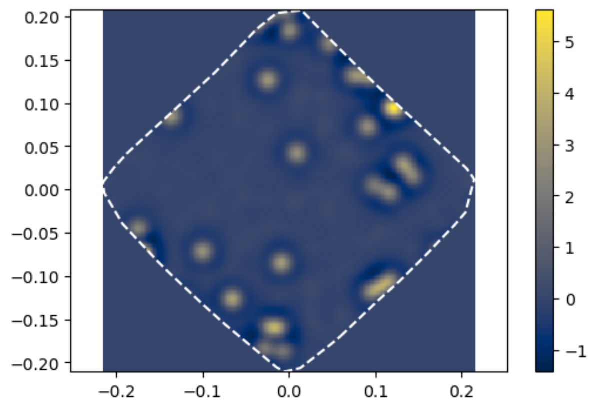


Same plot as above. The dashed boundary denotes the boundary of data points that are $\sim 2^\circ$ away from the father pixel.

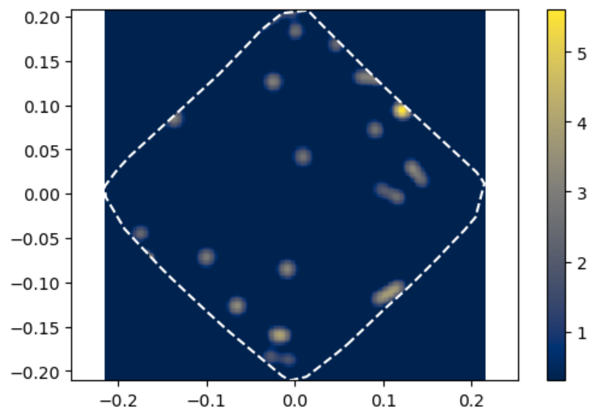


Example showing how the $\sim 2^\circ$ boundary is used to define the grid points that we use to calculate the wavelet map.

At this point, we can calculate the map of wavelet coefficients for this particular father pixel and set a threshold that picks out the point sources $(\mathcal{S}_0 > +0.3)$. The maps below correspond to first-order Mexican Hat wavelets with scale parameter $a = 0.6^\circ$ and a grid with 0.1° spacing.

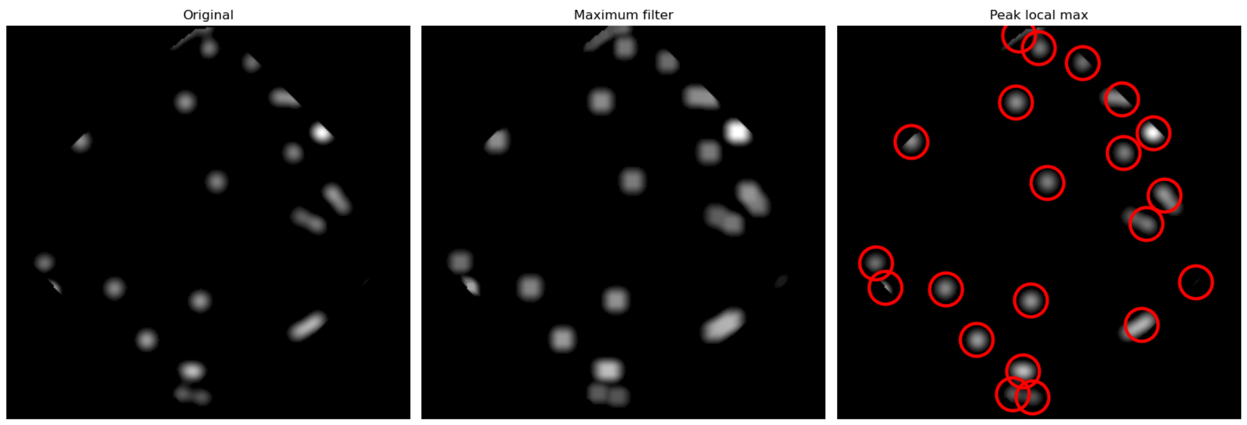


Raw wavelet coefficient map of father pixel npix = 58.

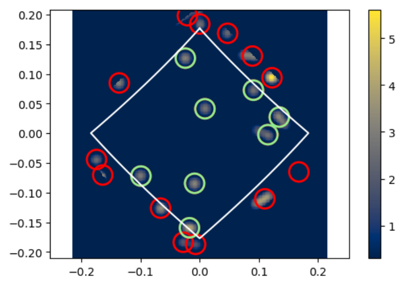


Wavelet coefficients father pixel npix = 58 satisfying the condition $\mathcal{S}_0 > +0.3$.

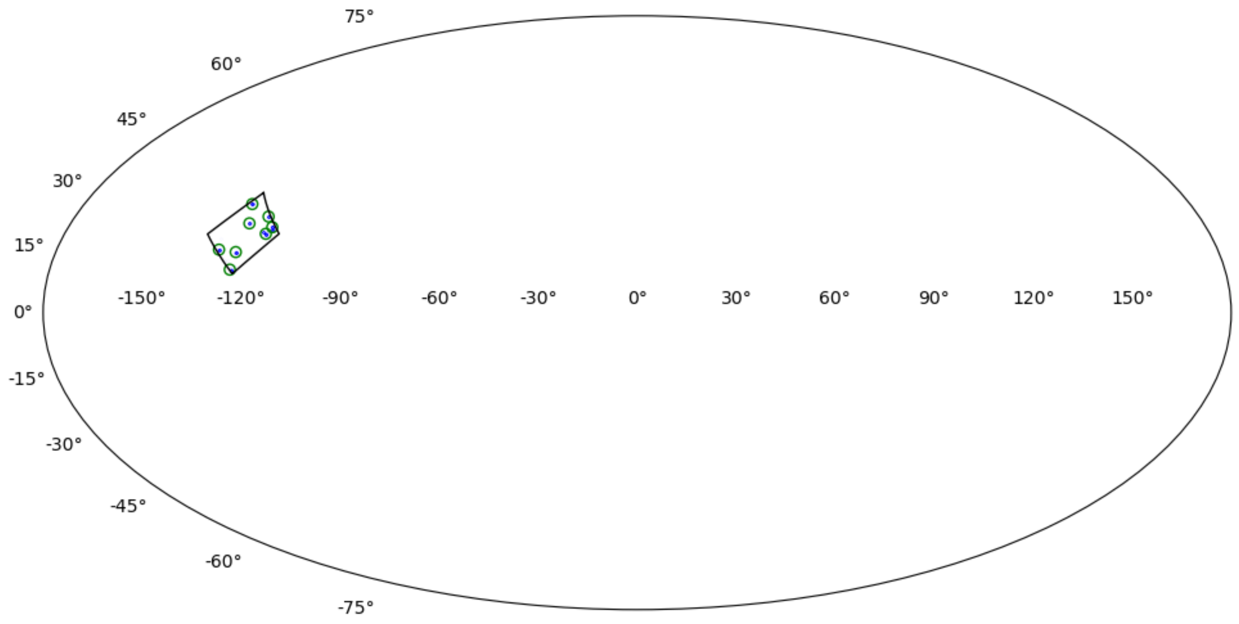
Applying the skimage.peak_local_max function to the thresholded map, we obtain the following set of identified point sources for the simplest settings of skimage.peak_local_max (i.e., *min_distance* $\sim 0.4^\circ$).



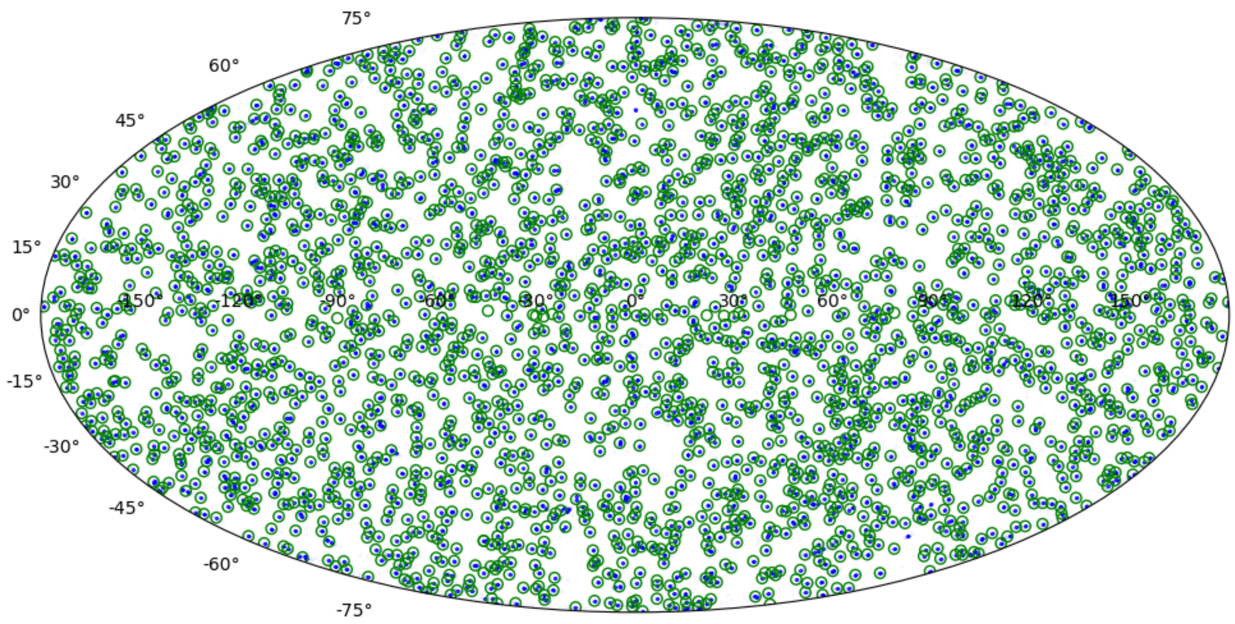
Then, we filter out those peaks that are outside the father pixel boundaries:



Finally, we project these peaks back to the sphere and compare their positions to those of the true point sources.



Performing this analysis for the entire map, we obtain the following list of candidate point sources:



As this is a very simple point source finder, there are several obvious optimization steps:

1. Optimize the scale and choice of Mexican hat wavelet for detection
2. Allow for detection of nearby point sources