Using 1D Sums to Find Fiducials

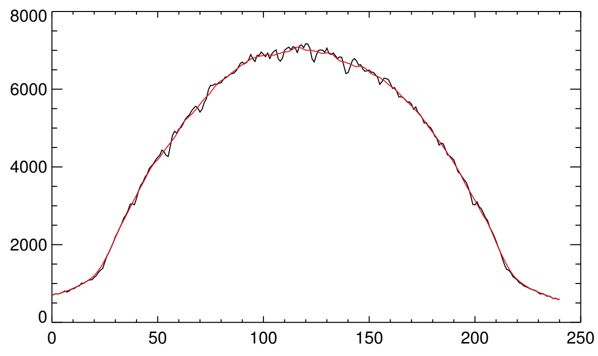
Jeren Suzuki

Last Edited August 28, 2013

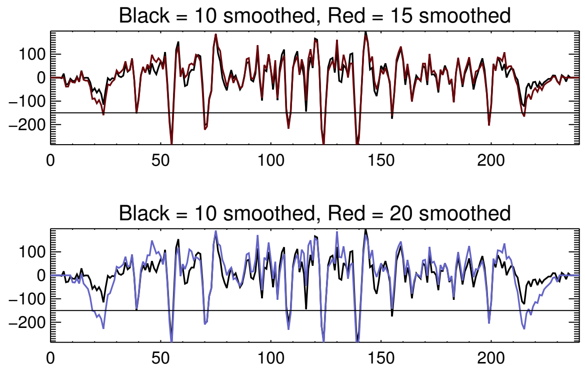
Contents

1 Introduction

Instead of using convolution filters, we look at 1D sums to identify fiducials.



(a) This is the 1D sum of a solar image. Small dips are seen in the profile corresponding to fiducials. The line in red is the sum smoothed by 10 pixels.

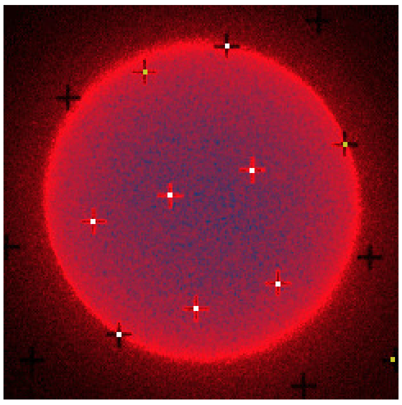


(b) We subtract the smoothed profile from the raw data to emphasize dips. Comparisons of the smooth amount change the width and number of the dips, although not really the depth.

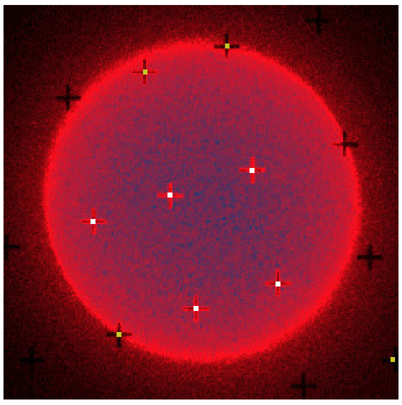
Figure 1

2 Finding Fiducials

For each position below a certain threshold (see Figure 1b) a row/column is returned. Once we have an array of possible fiducial row and column positions, they are matched against each other using a method that iteratively checks to see if a pair of coordinates is a fiducial. For Figure 2, the white squares are pairs where both X and Y coordinates were returned from thresholding, the yellow squares are pairs where one of the coordinates had to be guessed since it was not found.



(a) 10 pixel smoothed sum



(b) 20 pixel smoothed sum

Figure 2

There is a problem with the row/columns returned from thresholding the sum - smoothed sum; sometimes the number of indices don't match up. i.e., we "see" 7 fiducials in one axis but only 5 in another. How do we correctly rule out extra fiducials in one axis but not the other?

Also, this summing implementation aims to simplify and speed up the fiducial finding process, but once we have an array of X and Y positions of fiducials, verifying they actually lie on fiducials requires methods we tried to move away from.

2.1 Using Gaussian Fits

It turns out that using Gaussian fits to find fiducials is faster than expected. After finding all possible row/column combinations of fiducials, each candidate's 1D sum is fitted with a Gaussian, regardless of being a fiducial or not. We look at the χ^2/N statistic to order out fits by how 'fiducial' they are. Since we only need, say, M fiducials,

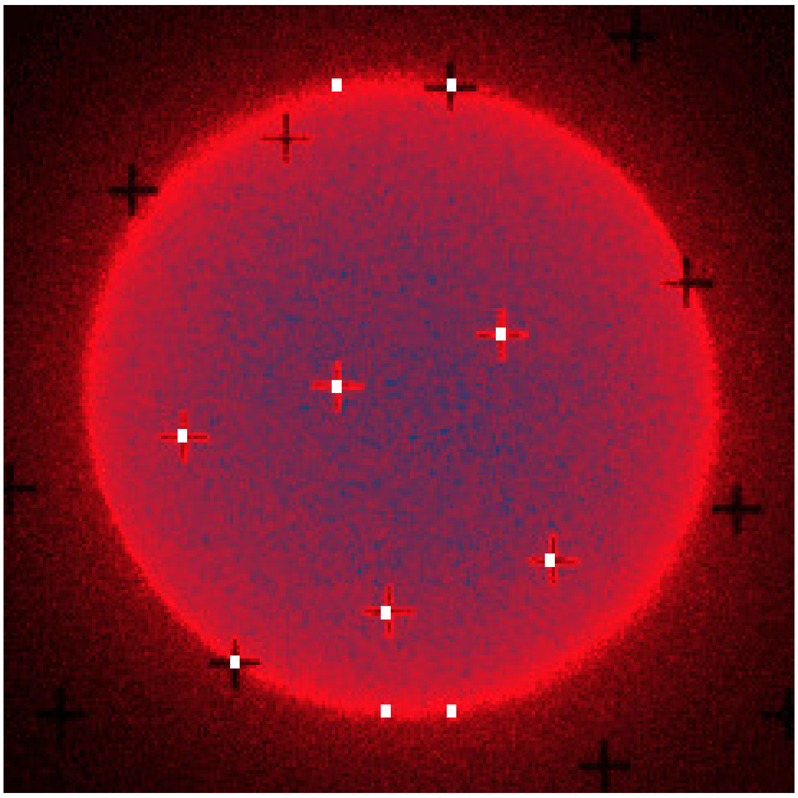


Figure 3: Gaussian fits with a $\chi^2/N > 20$

Table 1. Comparison of peak-fitting

X pos	Sub-pixel x (Parabolic Peak Fitting)	Sub-pixel x (Gaussian Peak Fitting)	Albert's x	Y pos	Sub-pixel y (Parabolic Peak Fitting)	Sub-pixel y (Gaussian Peak Fitting)	Albert's y
54	54.4053	54.4478	53.9459	108	107.946	108.044	109.100
70	70.3620	70.3049		39	38.7651	37.7765	
101	101.355	101.287	100.782	123	124.465	123.740	124.789
116	117.014	117.017	116.529	54	55.1857	54.8310	55.6844
136	135.756	135.700		215	213.553	217.142	
151	151.477	151.437	150.967	139	138.696	139.406	140.352
166	166.980	166.882	166.390	70	69.6979	70.3697	71.3244

we can sort the fiducials by their χ^2/N and pick the top 5.

We're walking a fine line here; instead of minimizing our χ^2/N, we're maximizing it.

2.2 1D Sums

1D Gaussians provide a robust method to identify fiducials but they take quite a long time to run. An alternative is to use another 1D sum to identify fiducial shapes in cropped areas defined by running a larger 1D filter. Figures 6 and 7 illustrate the relationship between the region we look at versus what the 1D sums look like. In Figure 5, 7 fiducials are found using the 1D sum method.

2.3 Sub-Pixel Fitting

Once I know where the peaks are in the arrays of Figure 7, I fit a parabola to the 3 pixels encompassing the peak maximum in each direction.

3 Comparison of Fiducial Finding Methods

Table 1 compares the Gaussian and parabolic-peak fitting results to Albert's fiducial positions.

```
>> help,*(bbb[0])
```

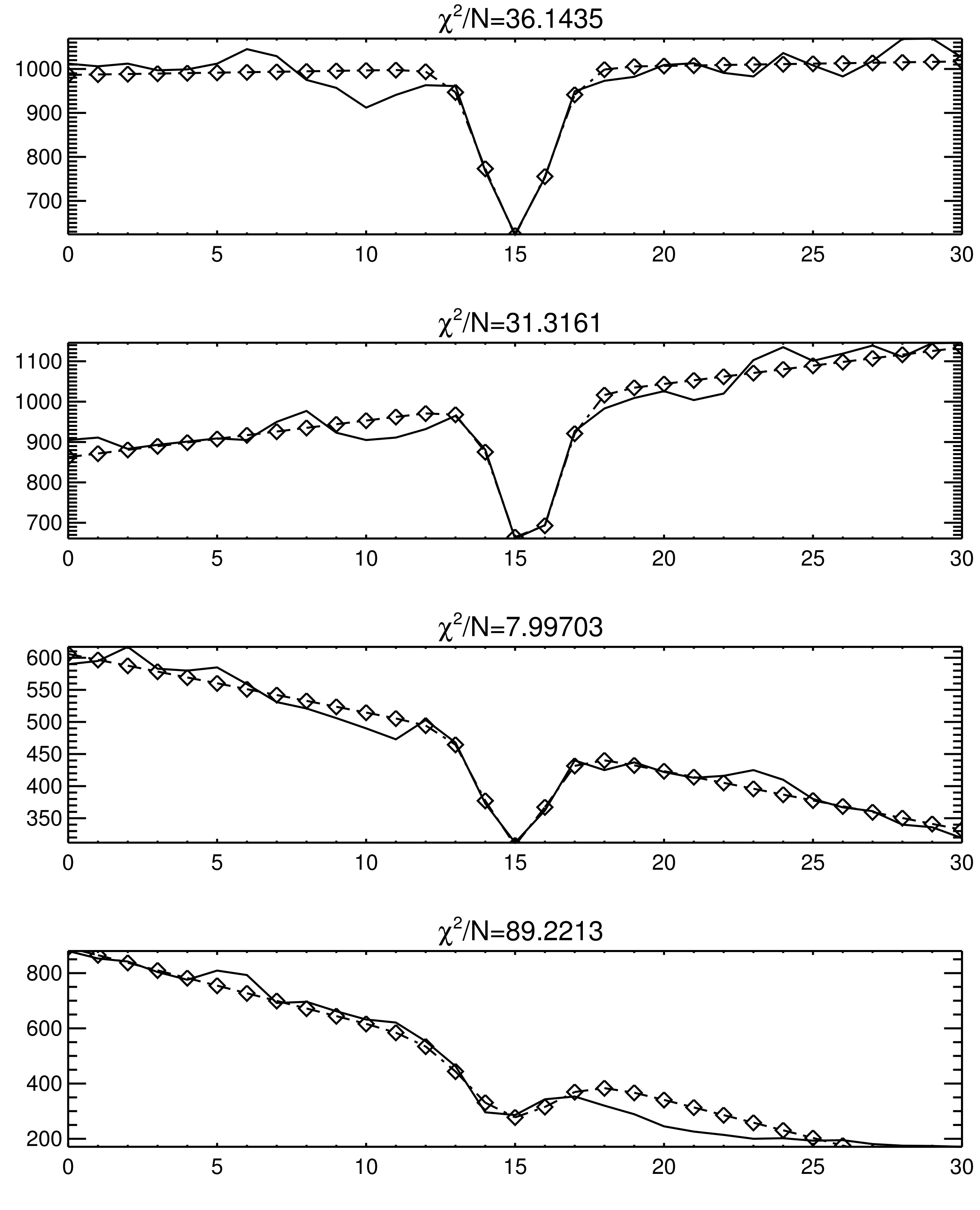


Figure 4: These are all real fiducials, ranging from looking nice (first two) to being on the solar limb (last 2).

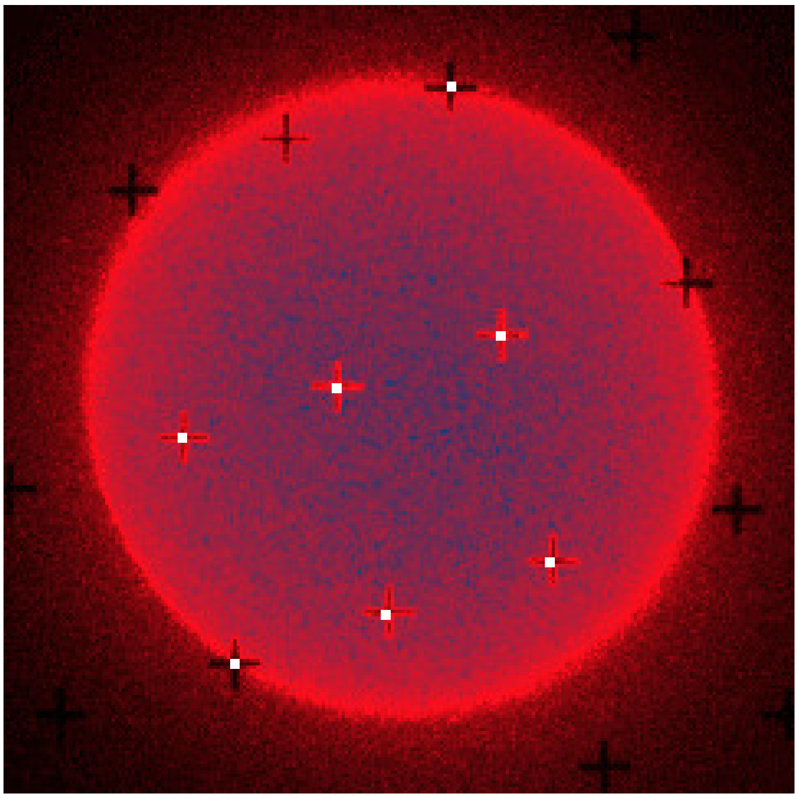


Figure 5: Fiducials found using a 1D summing method on the entire image followed by another 1D summing method on a cropped region based on fiducial candidates. Using a threshold of 100 on an `smooth(array,10) - array` identifies the white squares.

```
** Structure <260a348>, 2 tags, length=180, data length=178, refs=1:
   REG INT 1
   FIDARR STRUCT −> FIDPOS Array[11]
>> help,(*(bbb[0])).fidarr,/str
** Structure FIDPOS, 4 tags, length=16, data length=16:
   X FLOAT 50.0000
   Y FLOAT 132.000
   SUBX FLOAT 50.8438
   SUBY FLOAT 133.291
```

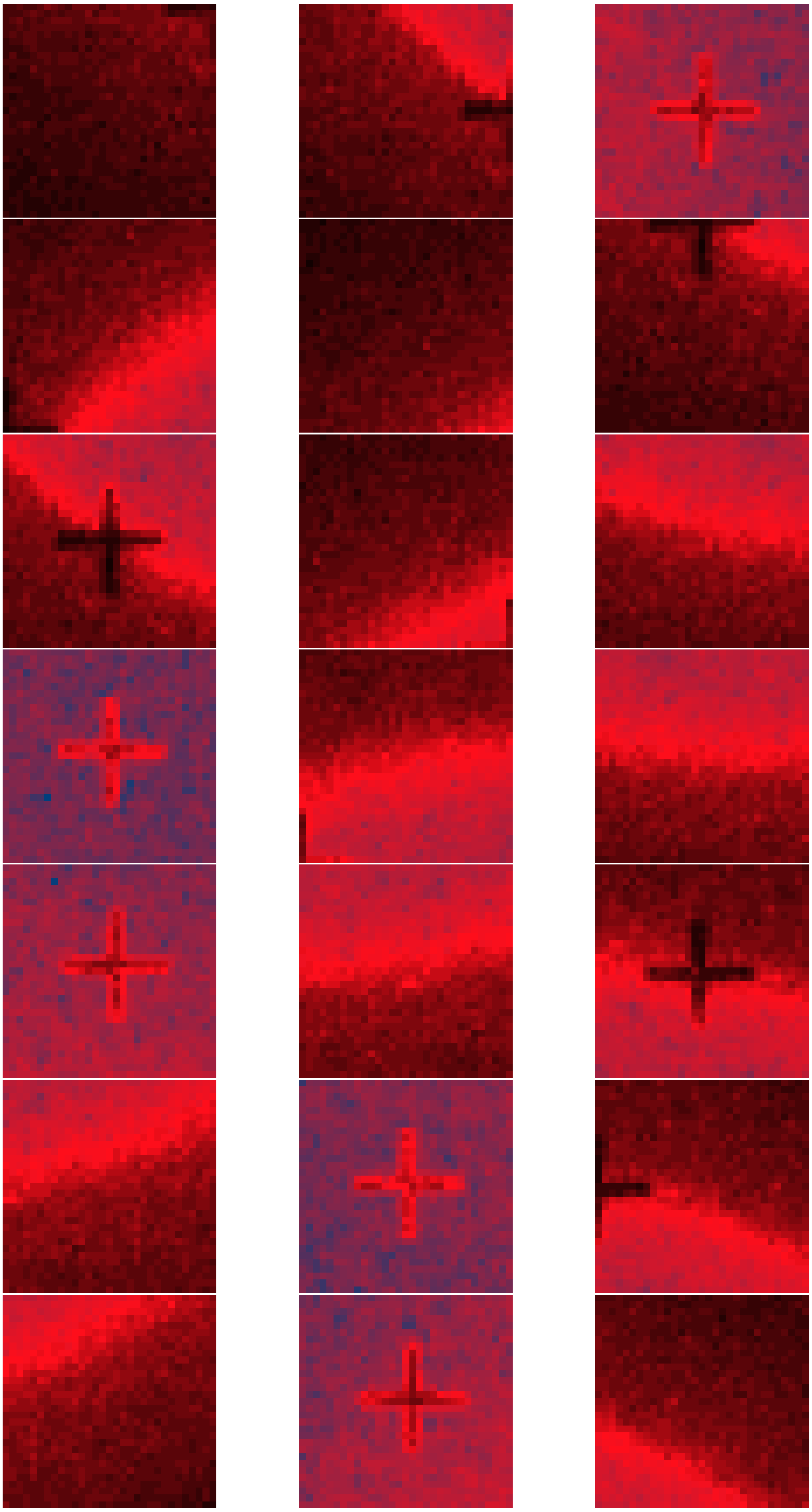


Figure 6: Each possible fiducial candidate

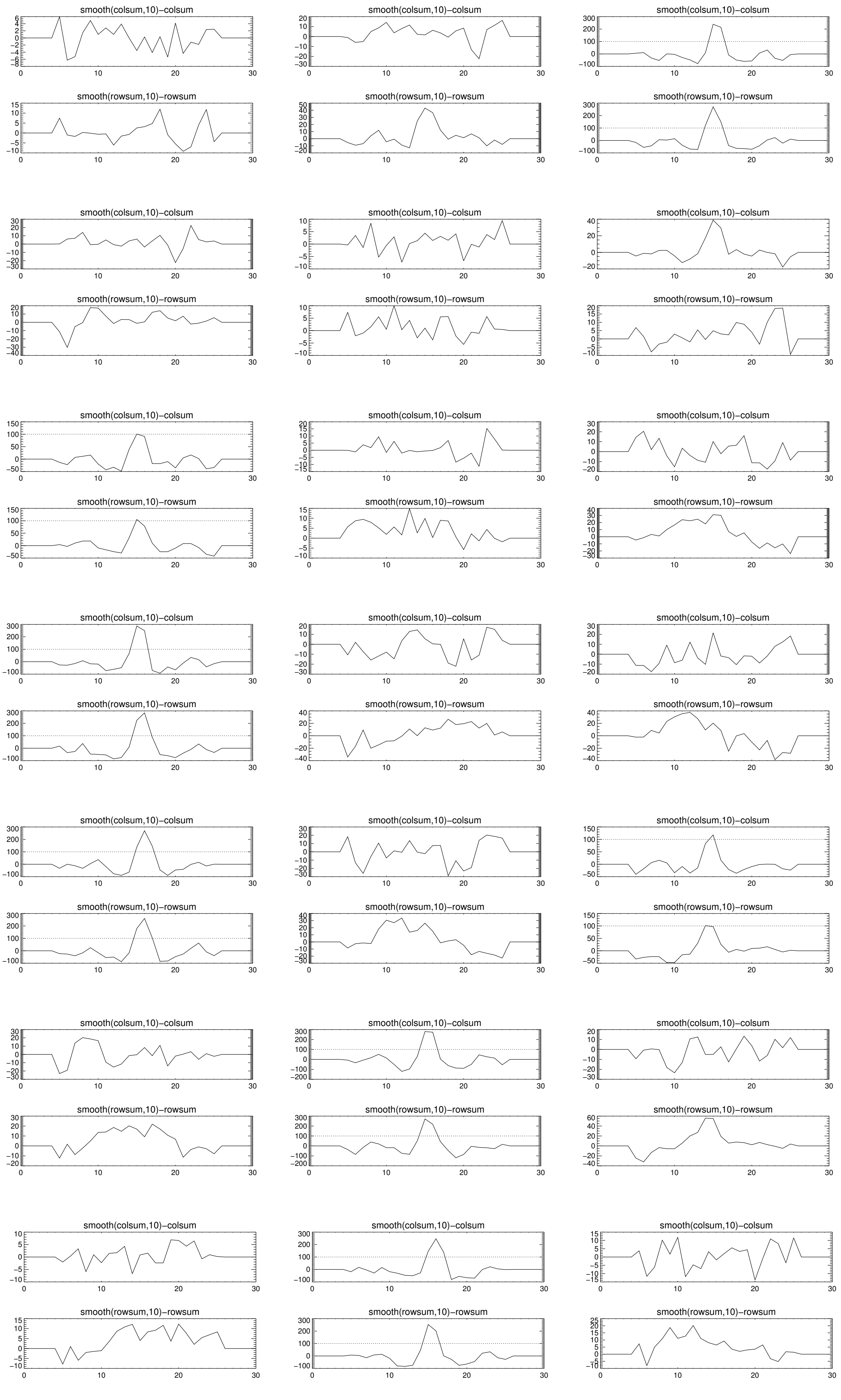


Figure 7: The horizontal line is at 100; if there are any elements of the array above 100 for both a column 1D sum and a row 1D sum, then the cropped area is identified to have a fiducial in it.