Code Explanation

Jeren Suzuki

Last Edited August 27, 2013

Contents

Introduction

This pdf is meant to illustrate what each step of the code is doing. The code is split up into four major parts. Preparing parameters, thresholding image to find suns, limb-fitting whole suns, and finding fiducials.

1 Preparing Parameters

1.1 Loading Image

This is the starting image we use to analyze for sun centering and fiducial finding.

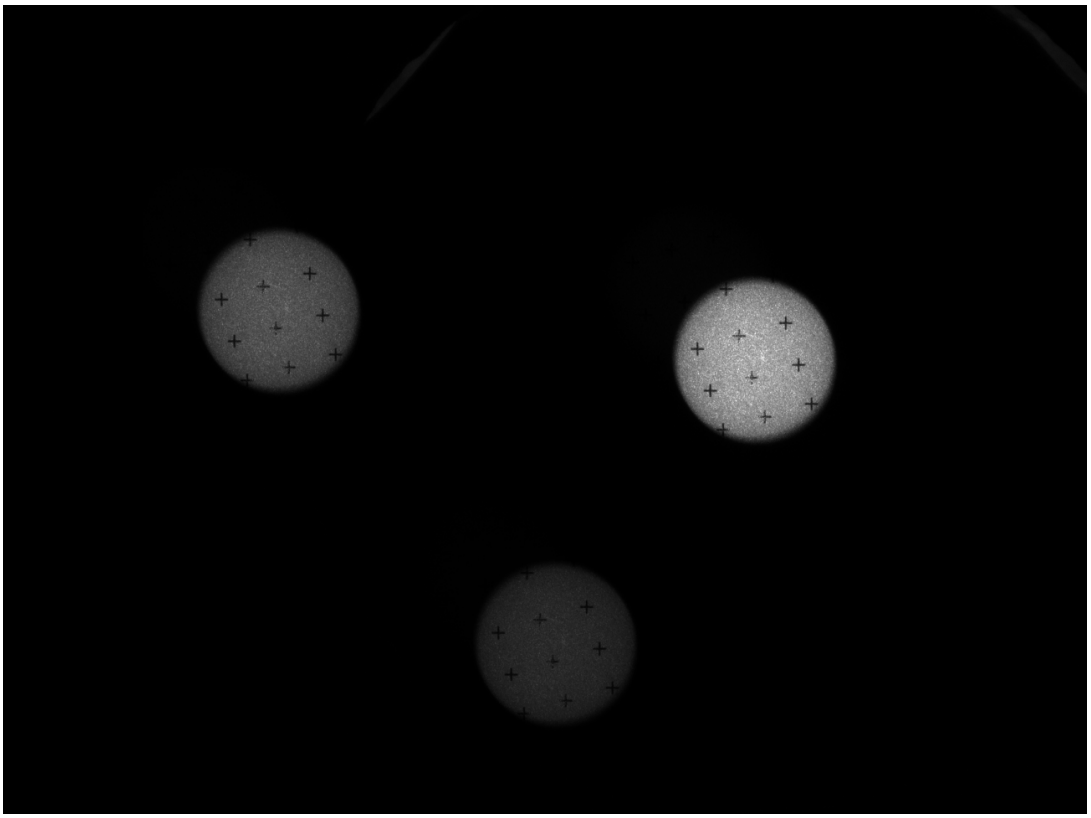


Figure 1

1.2 Reading In Parameter Block

```
1  scan_width 10                ; Distance to next chord when picking chords to limb-fit
2  sundiam 70                   ; Approx Solar diameter, deprecated
3   nstrips  5                  ; Number of pairs of solar chords to limb-fit per direction
4   ministrip_length  4         ; Length of limb profile to linear fit
5  crop_box 120                 ; Half-width of box used to find fiducials in
6  elim_perc 1                  ; Percentage of highest pixels to eliminate when finding threshold
7  n_smooth 900                 ; Elements to smooth by when finding threshold
8  border_pad 50                ; If solar center is within this value of border, marked as a partial sun
9   triangle_size  .25          ; Percentage of image height to use for triangle sides for making clipped-bottom-corner
       mask
10 fid_smooth_thresh -150       ; Threshold to determine row/column positions of fiducials
11 onedsumthresh 80             ; Once looking at fiducial candidates, look at 1D sum of smaller fiducial crop and
       threshold difference of smoothed array - original array by this
12  disk_brightness  15         ; Arbitrary pixel brightness to eliminate bright fiducial candidates which are on the
       solar disk but are not on a fiducial
13 fid_crop_box 15              ; Half-width of box used to analyze fiducials
14 fid_smooth_candidates 15     ; Smoothing paramater for 1D sums of fiducial candidates
```

2 Thresholding Image

After we read in the necessary parameters, we turn our attention to thresholding the image to find the centers of the suns. Starting with the initial image, we sort the image according to pixel values and see a grouping of differently dimmed suns.

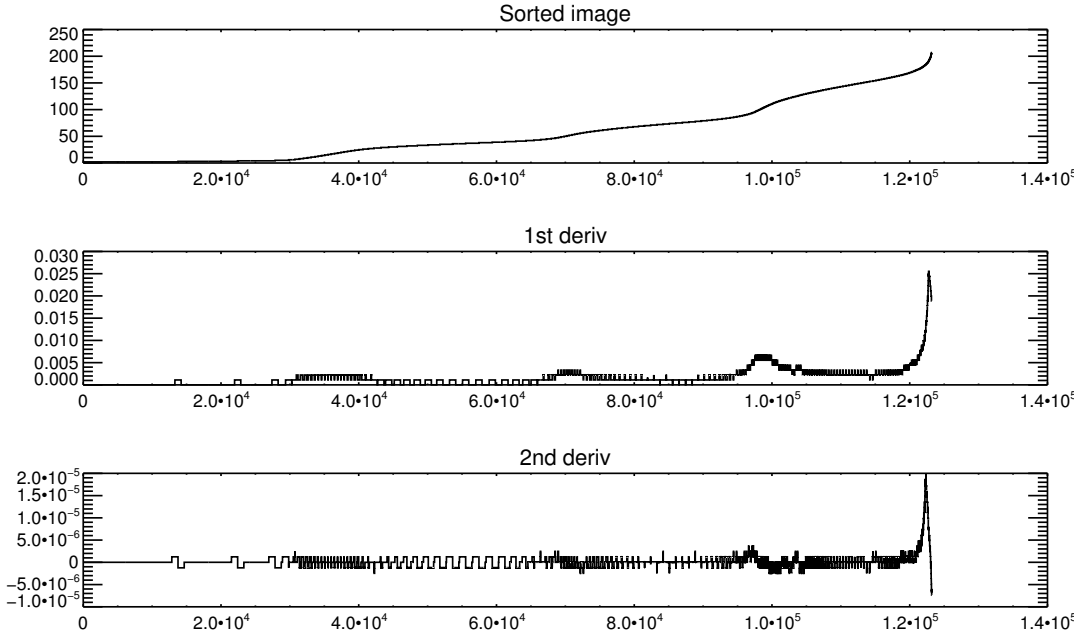
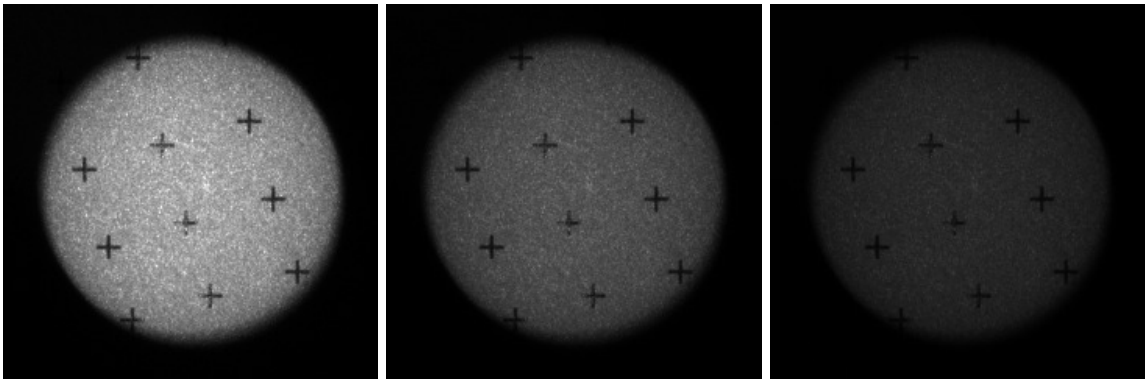


Figure 2: We look at the second derivative because it consistently quantifies the thresholds to identify each solar region by. With the right-most part of the 2nd derivative zeroed out, we identify the three peaks corresponding to what pixel value we should threshold by to separately identify each sun.

Using these three thresholds, we create separate masks, one for each sun.

2.1 Quick mask centering

Now that we have three cropped regions, we have to identify what brightnesses they are. We do this by looking at a low enough threshold that all the solar pixels for every brightness are included in the mask but not any of the background pixels. We then use IDL's `LABEL_REGION` to assign a label number to each



(a) 100% brightness region, henceforth Region 1 (b) 50% brightness region, henceforth Region 2 (c) 25% brightness region, henceforth Region 3

Figure 3

shape. A shape is defined to be a mask where there are adjacent pixels. If done correctly, we end up with 3 shapes that have now had their values replaced with what index they have been assigned. For this image, we have three regions, one a mask of 1s, one a mask of 2s, and one of 3s. Now we use `HISTOGRAM` to bin these shapes and thus their locations. For each mask, we calculate the average value of the pixels from the starting image and

We find the centers of each cropped region in fig. 3 using a simple masking method. With the centroid determined by a simple masking method, we have to make sure that the center is from a whole sun. The process so far has not taken into account any spatial information so the next step is to look at a border region of the mask.

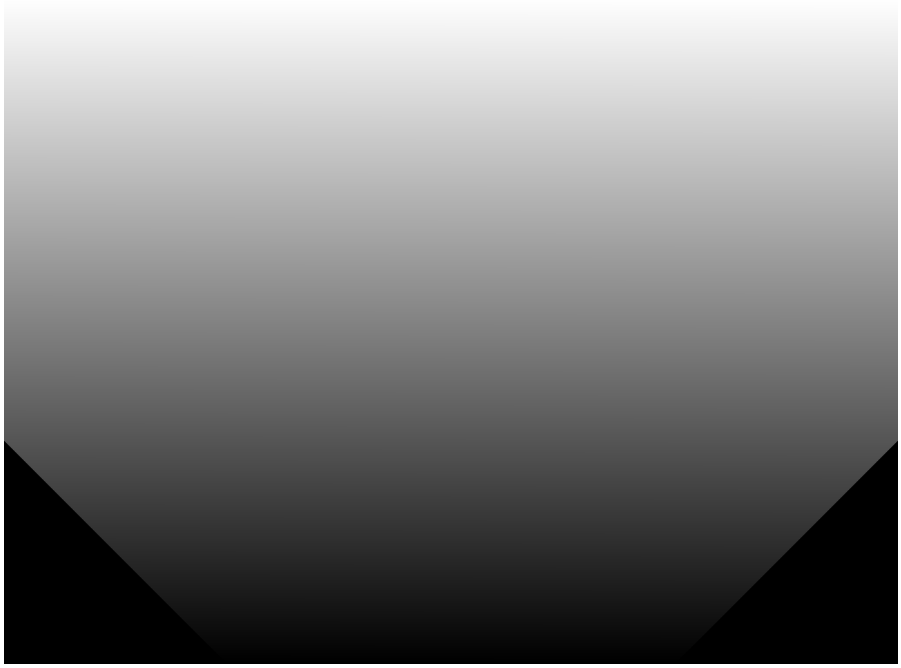
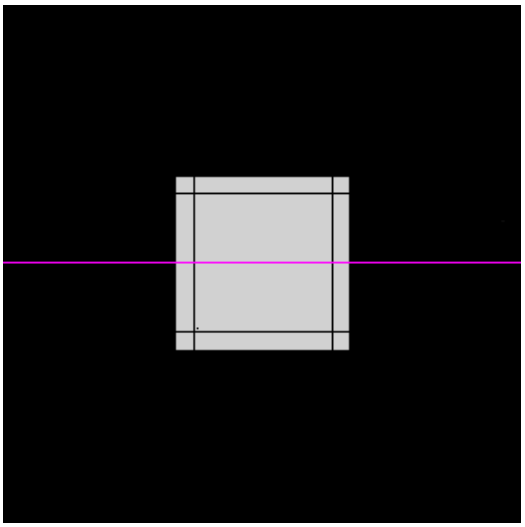


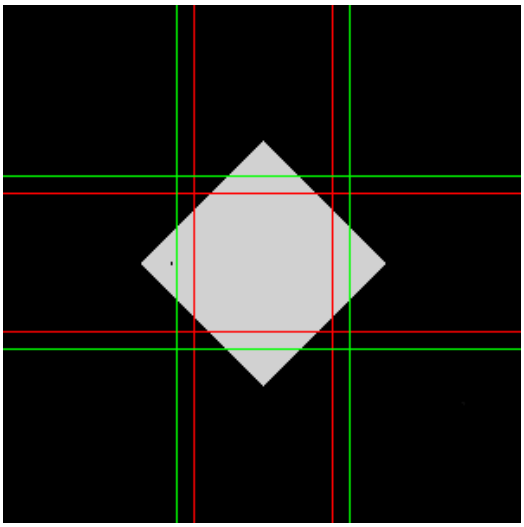
Figure 4: What our mask should look like - side of black triangle is 1/4 of image width

To check if the sun is a partial sun, we calculate the distance of the measured center to the edges of the image. If the center of the sun is too close to the edge, then there must be a part of the sun that is cut off in the image. This method works simply for a rectangular mask but our mask is not so simple.

The bottom corners of our mask are cut off, essentially making it so that that part of the mask will never see any rays of light. See fig. 4. To deal with the distance from the edges of the bottom corners, we rotate the coordinates of the center by 45 degrees.



(a) The black lines represent 10% of the width/height and the hypothetical solar center is the black dot in the lower left-hand corner. As you can see, by this simple check's standards, the center is OK (but it's actually in the corner which is bad).



(b) The green lines outline the shape of the box in fig. 5a while the red lines outline the shape of the box 10% from the edges. By rotating the black dot (or in this case, the gray square) and checking if it is within 10% of the edge, we can essentially check the distance from the diagonal corner borders.

Figure 5: By combining two edge checks, one with 45° rotated coordinates, we can quickly calculate the distance from the edge of the mask.

3 Limb Fitting

We cut 5 chords centered around the masked center position in both axes (See fig. 6a). With the same threshold found in section 2, we look at where the chord profile crosses the threshold (See fig. 6b). We use a four-pixel limb profile with 2 pixels above the threshold and 2 pixels below the threshold. We then do a linear fit on these four pixels to see where it crosses the threshold. With a pair of limb-fitted threshold crossing points, we create a chord and find it's center. Doing this 5 times, we average the positions of the chord centers to be the limb-fitted center position.

4 Finding Fiduials

Now with limb-fitted centers, we re-crop the image to work with (It shouldn't look much different than fig. 3 if we masked the centers correctly).

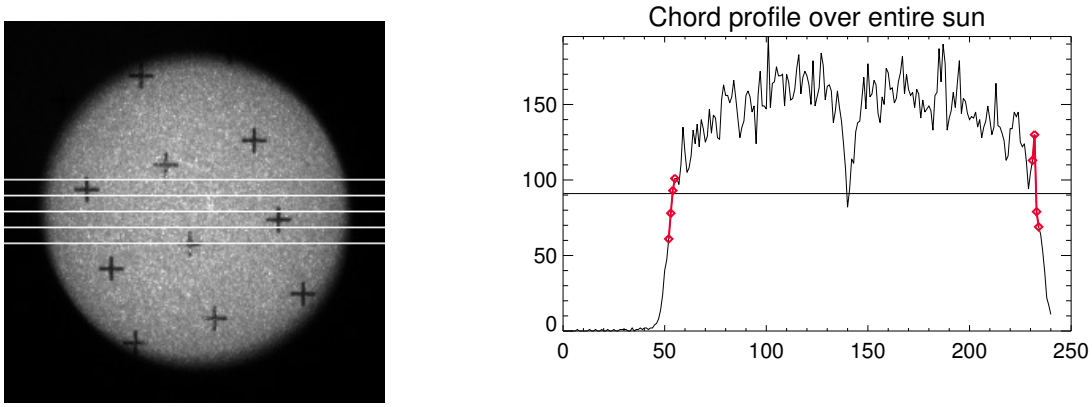
4.1 First 1D Sum

To find fiducials, we calculate a 1D sum in one direction and look for dips in the array corresponding to fiducials.

After thresholding the difference of the 1D sum from it's smoothed counterpart, we identify the column/row positions of fiducials.

4.2 Ruling Out False Fiducial

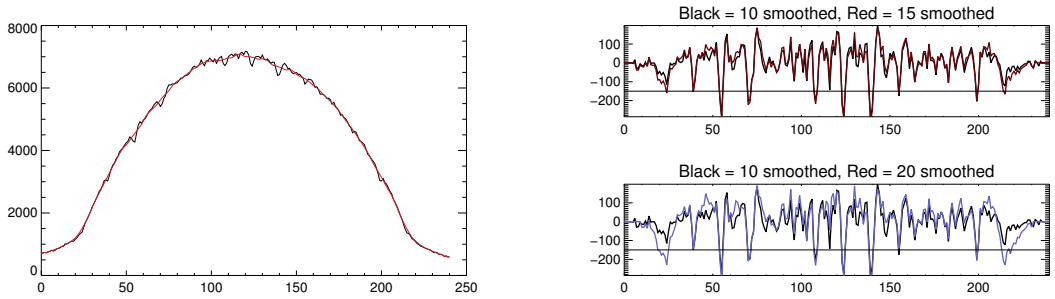
With a bunch of x and y locations of fiducials, we have to determine which pairs of coordinates are actually fiducials. The first step is to look at each possible pair and eliminate those that are too bright.



(a) The middle chord passes through the center of the sun

(b) The indices in red are the pixels we apply a linear fit to see where it crosses the threshold (which is the horizontal line).

Figure 6



(a) This is the 1D sum of a solar image. 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. It says "Red" in the title, but it's really blue. Typo.

Figure 7

This affects candidates which are located on the solar disk but do not lie on fiducials. See fig. 8.

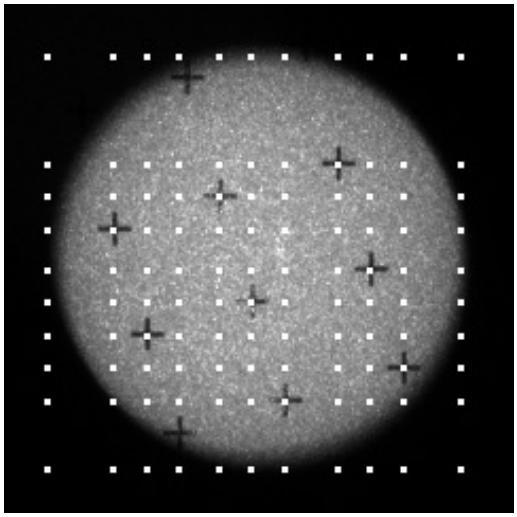
Next, for the remaining candidates, we look at the 1D row and column sums to see if there are any obvious dips. Figure 9 is a fiducial candidate with obvious dips in the row/column arrays. Figure 10 on the other hand, has a distinctive shape for 1D sums and does not have any obvious dips. Using another thresholding technique, we smooth the 1D sum and look for peaks in the difference of the smoothed 1D sum minus the 1D sum. If there are any pixels above this secondary 1D sum threshold, then the fiducial candidate is considered a bona-fide fiducial.

4.3 Sub-pixel Fiducial Positions

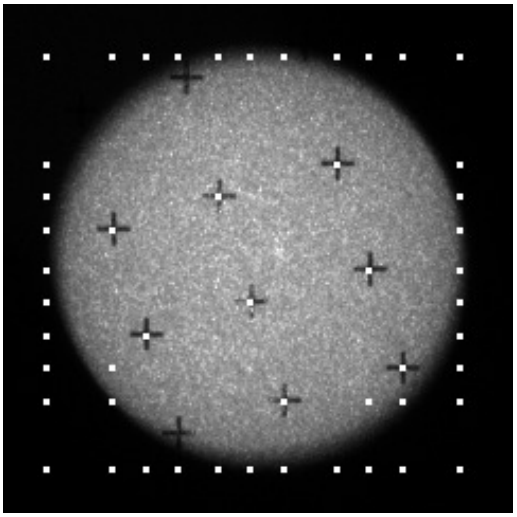
Now that we have a column/row position of a fiducial, we look to find a sub-pixel location of the fiducial. Now, this technique has proven to be more efficient in past iterations of the program and is still under construction.

In earlier versions of the program, we used a 2D convolution filter with a kernel that looks like a fiducial. Areas on the image that look like fiducials have high correlation values and thus peak in the 2D convolved array. Once we had computed the local maxima (which gave us rough fiducial positions), we used 2 1D parabolic fitting procedures to the peak of the local maxima. The location where the two parabolas intersect (since they're perpendicular to each other) is the sub-pixel parabolic fit for the fiducial position.

In this version of the code, we don't have any natural peaks in the 2D image so we have to look for peaks elsewhere. Thankfully, we have peaks in the 1D sums. Instead of using a parabolic fit to the 2D

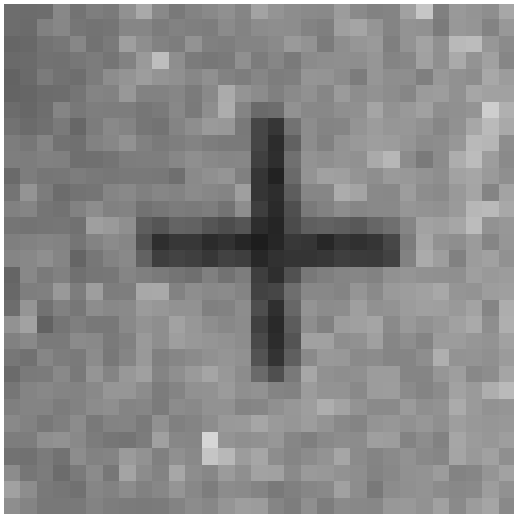


(a) All possible positions of fiducials according to first 1D sum and matching up column/row positions.

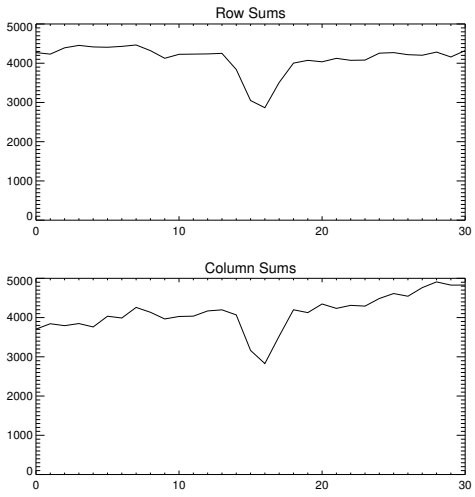


(b) Ruling out values above a certain threshold

Figure 8



(a) Cropped region of a fiducial candidate



(b) 1D sums of cropped region

Figure 9

convolution of the image, we fit a parabola to the peak of the 1D sum. Same deal, we look where the two parabolas intersect. I'm honestly not sure if I can do this but for now this is how we're doing it.

After thresholding the peak of the fiducial candidate's 1D sum, we end up with fig. 11.

5 Final Words

Table 1 is an example of how the final data structure is kept. Since the number of fiducials cannot be predetermined and I don't know how to append structure elements within a nested structure, the fiducial positions are kept in a separate pointer structure which takes the form of table 2. The structure of the `FIDARR` array is shown in table 3.

Table 1. Final data structure of solar region

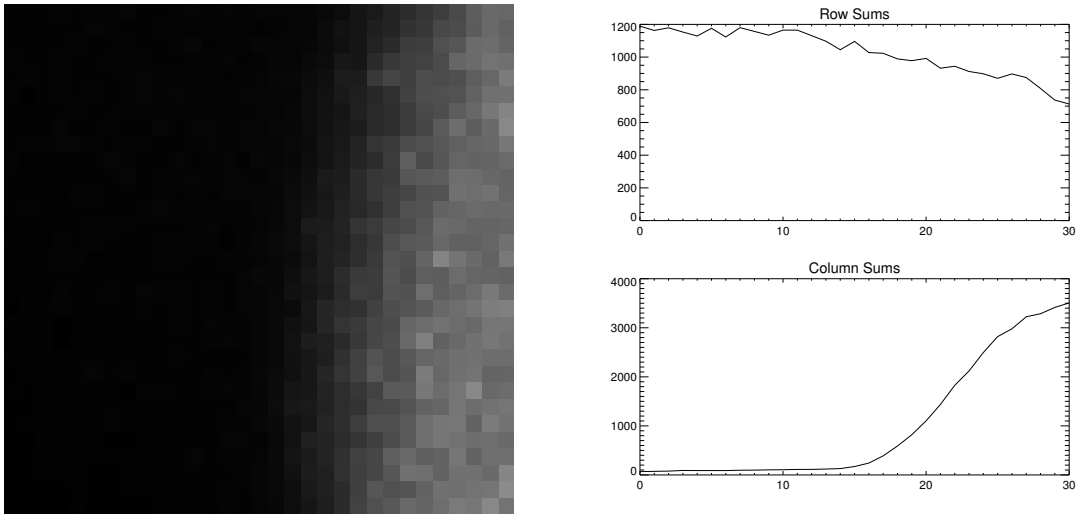
Name	Type	Value	Notes
XPOS	FLOAT	875.802	Rough calculation using a simple masking method
YPOS	FLOAT	544.078	”
REG	INT	1	Region ID #: 1 is 100%, 2 is 50%, 3 is 25%
THRESH	FLOAT	106.000	Threshold calculated from sorting array and taking derivatives. Used in both finding rough X-Y center as well as the threshold for limb-fitting.
PARTIAL	FLOAT	0.	Flag that determines if the solar region is cut off on the edge or not. 0 means that it is not cut off
XSTRIPS	STRUCTURE	-> WHOLEXSTRIPS Array[5]	Strucutre containing the strips of whole solar data bound by a cropped region chosen by XPOS and YPOS
YSTRIPS	STRUCTURE	-> WHOLEYSTRIPS Array[5]	”
LIMBXSTRIPS	STRUCTURE	-> LIMBXSTRIPS Array[5]	LIMBSTRIPS contains a pair of arrays, ENDPOINTS and STARTPOINTS that mark the limbs of each strip of data from X/YSTRIPS
LIMBYSTRIPS	STRUCTURE	-> LIMBYSTRIPS Array[5]	”
LIMBXPOS	FLOAT	898.586	Center calculated from LIMBXSTRIPS
LIMBYPOS	FLOAT	541.795	”
NPIX	FLOAT	26680.0	Number of pixels above threshold. Used to determine if the sun was a partial, deprecated.

Table 2. Structure of pointer containing fiducial positions

Name	Type	Value	Notes
REG	INT	1	Region ID #: 1 is 100%, 2 is 50%, 3 is 25%
FIDARR	STRUCT	-> FIDPOS Array[11]	Structure containing the fiducial positions and sub-pixel positions

Table 3. Structure of `FIDARR` structure

Name	Type	Value	Notes
X	FLOAT	51.0000	X position found through first set of 1D sums
Y	FLOAT	133.000	”
SUBX	FLOAT	51.8438	X position found with parabolic peak-fitting of second 1D sum
SUBY	FLOAT	134.291	”



(a) Cropped region of a fidcuail candidate

(b) 1D sums of cropped region

Figure 10

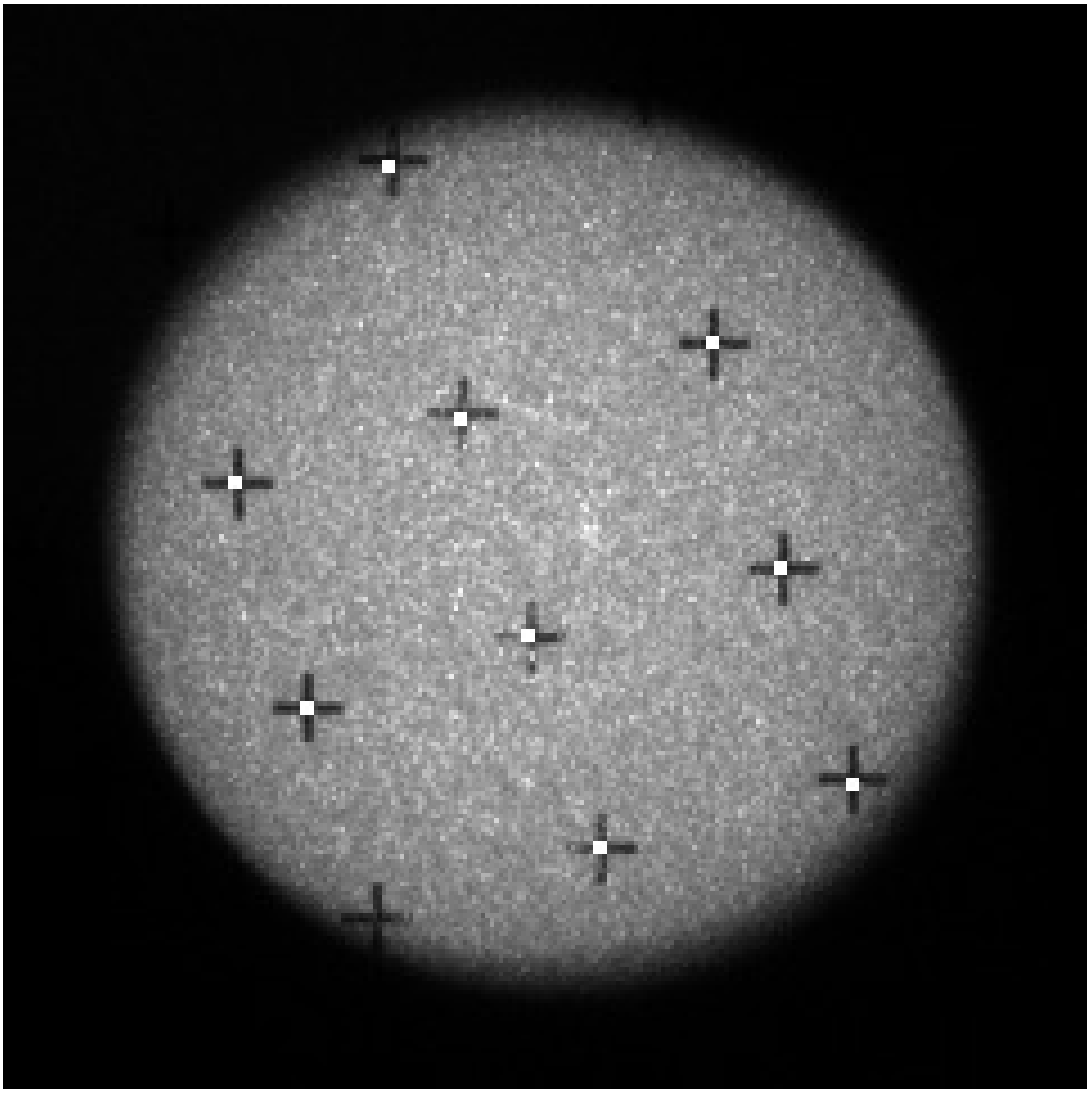


Figure 11: Final set of fiducials