Final Documentation Draft

Jeren Suzuki

Last Edited June 25, 2013

Contents

1 Introduction 1

2 Introduction cont. 1

3 Setting Up Parameters 2
3.1 Possibility for Error 3

4 Prepare Image for Limb Fitting 3

5 Fitting Limb Strips 4

6 Finding Fiducials 5
6.1 How Robust is Fiducial Finding? 6

1 Introduction

Starting with an image of at most three suns on a fiducial grid, we find their centers and relative position to fiducials (which provide a physical distance calibration). With a 1296 x 966 image with three suns, the program from start to finish takes approximately .15 seconds.

2 Introduction cont.

Code steps:

1. Load Image
2. Load parameters from pblock.txt
3. Sort image and cut off top .1% of pixels (top 1% was actually too much)
4. Smooth, take deriv, smooth again, take deriv again of sorted array, find peaks that correspond to where the values of different solar regions change. These peaks are used to threshold different solar regions.
5. Mask image above thresholds to find centers of every shape (sometimes it's not a circle)
6. If the centroid of the shape is within a certain distance to the image edge, mark as partial and cease further analysis
7. Crop remaining whole suns
8. Extract 5 strips around centroid for both X and Y direction
9. Extract a pair of limb strips for each whole strip
10. Applt linear fit to limb strips
11. Use new threshold-crossing positions of fit to calculate chord lengths
12. Average midpoints of chords to find limb-fitted centers
13. Analyze the cropped image for fiducials
14. Using the fiducial positions, compare solar positions to a position defined by the physical setup.

This is the form of the fiducial structure containing the positions and sub-pixel positions of fiducials for each solar region.

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

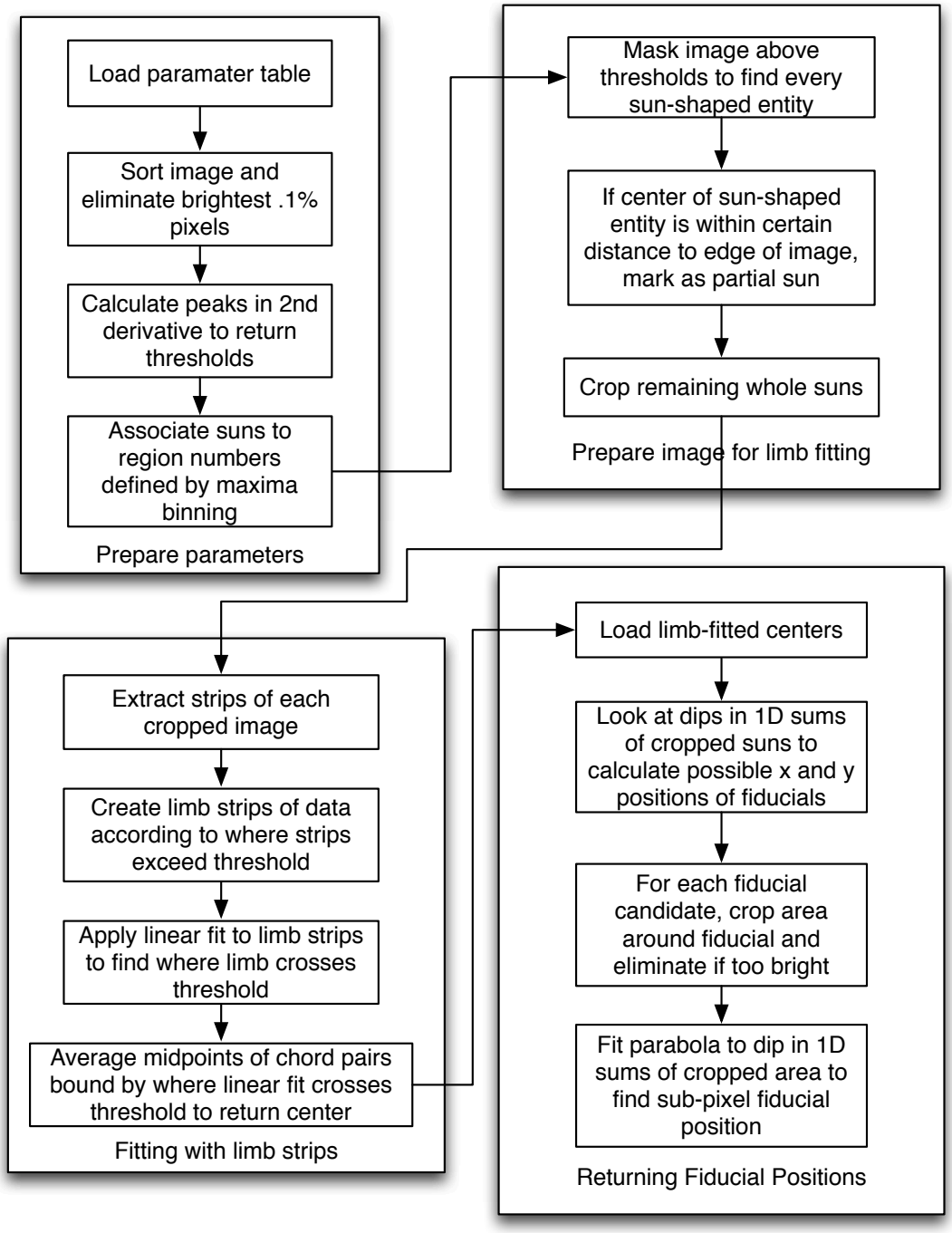


Figure 1

3 Setting Up Parameters

Before we analyze the solar image, we load a parameter table and assign values.

```
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
```

Table 1. Final data structure of solar region

Name	Type	Value	Notes
XPOS	FLOAT	210.522	Rough calculation using a simple masking method
YPOS	FLOAT	166.702	”
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	210.710	Center calculated from LIMBXSTRIPS
LIMBYPOS	FLOAT	167.172	”
NPIX	FLOAT	26680.0	Number of pixels above threshold

```
n_smooth 900                ; Elements to smooth by when finding threshold
border_pad 50               ; If solar center is within this value of border, marked as a partial sun
triangle_size .25           ; Percentage of image height to use for triangle sides for making clipped-bottom-corner mask
fid_smooth_thresh -150      ; Threshold to determine row/column positions of fiducials
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
disk_brightness 15          ; Arbitrary pixel brightness to eliminate bright fiducial candidates which are on the solar disk but are not on a fiducial
fid_crop_box 15             ; Half-width of box used to analyze fiducials
fid_smooth_candidates 15    ; Smoothing paramater for 1D sums of fiducial candidates
```

In Figure 2 we dynamically set thresholds for our image. Each bump in the top-most plot corresponds to a certain sun in a multi-sun image. By figuring out where the bumps change, we can threshold multiple suns for identification. We can't simply order them by brightness because we may have a 100% and 50% brightness sun in the same image and in order to determine the difference between a 100%/50% sun combo and a 50%/25% sun combo we must look at the actual threshold values.

3.1 Possibility for Error

Herein lies the problem of determining solar regions based on brightness and threshold values. We don't have a clear way of determining whether an image is dimmed before we even take a picture. If this happens, an image with a 100% and 50% sun may appear to be a 50% and 25% sun. There is no way to tell in our code, but hopefully we'll know enough about the time of day and pointing direction to account for this situation.

4 Prepare Image for Limb Fitting

After we set our threshold and parameters, we iteratively find and crop suns in the image. To determine if the sun is partially cut off, we create a border around the edge of our (usable) image. If the center is within the border, it is counted as a partial sun and excluded from further analysis. I emphasize *usable* because the image's bottom two corners are cutoff, reducing the usable part of the image. Figure 3 gives and example of what our mask might look like. The gradient is unimportant, it's purpose is only to emphasize the shape of the mask.

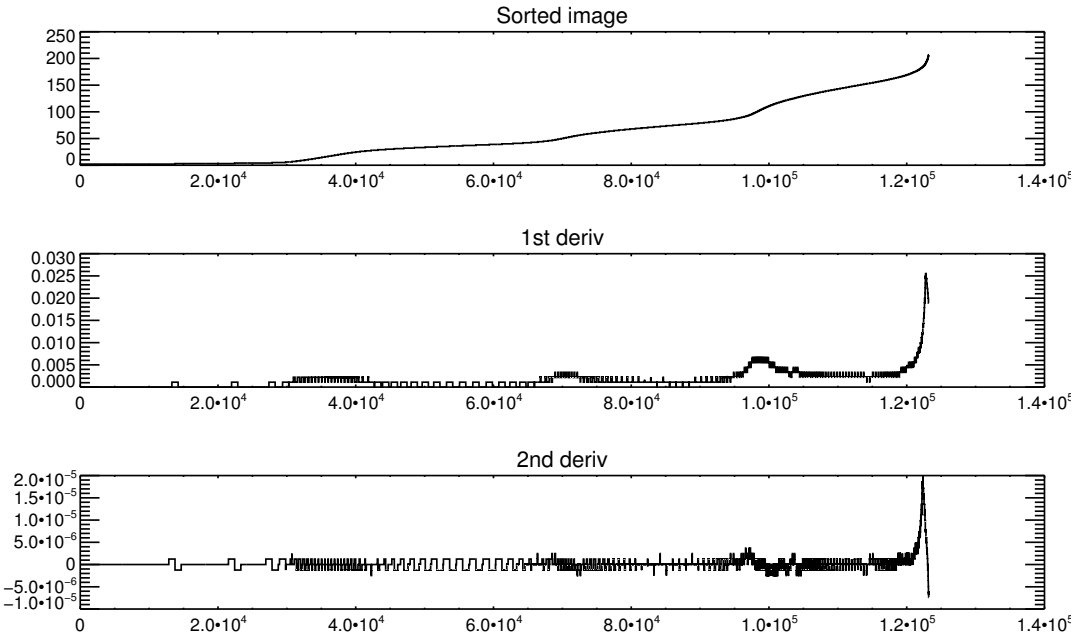
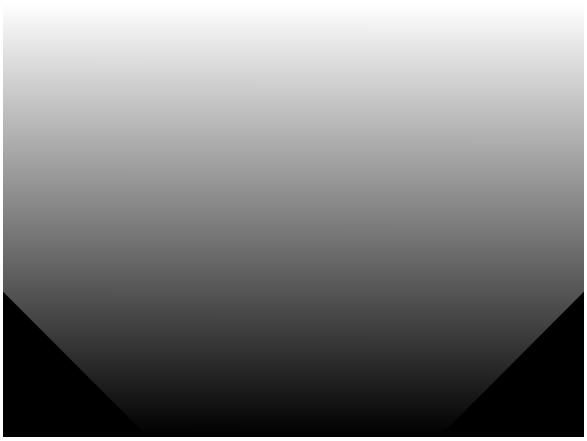
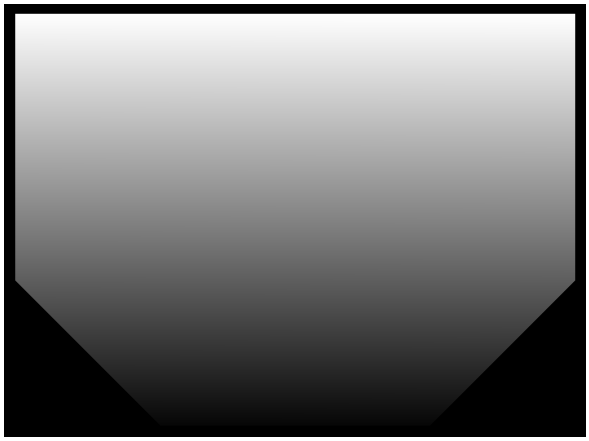


Figure 2 We look at the second deriv because it consistently quantifies the thresholds to identify each solar region by. Unfortunately (or for better), we repeat this process per image we analyze. In terms of total time spent from starting the program to returning limb-fitted centers and fiducials, setting thresholds takes up about 25%.



(a) What our mask should look like - side of black triangle is 1/4 of image width



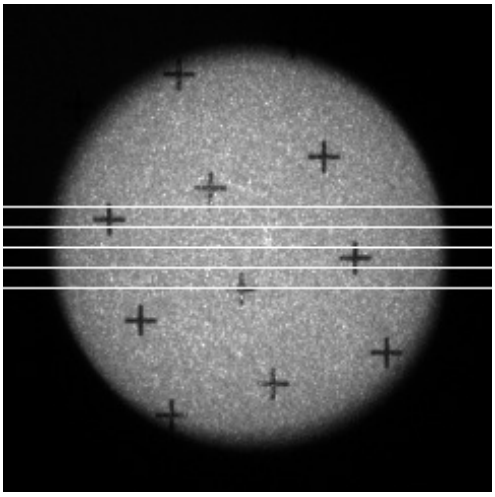
(b) A proposed mask that looks within a certain distance form the border.

Figure 3 The bottom corners will never see any data; the border mask takes into account the distance from the hypotenuse of the bottom corners

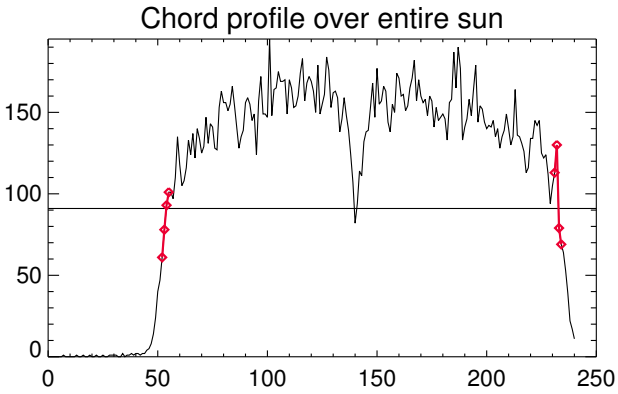
5 Fitting Limb Strips

In order to fit limb strips, we first identify chords of the sun to take strips of. First, we look at the center of the sun and at regular intervals, choose rows/columns centered around the X/Y center position. The result are chords as seen in Figure 4a.

Now looking at each chord, we select 4 pixels around where the chord crosses a threshold; 2 pixels below the threshold and 2 pixels above. See Figure 4b. One caveat is that if a fiducial is on the limb, our results may be skewed. This has been a persistent issue that we have not addressed fully yet.



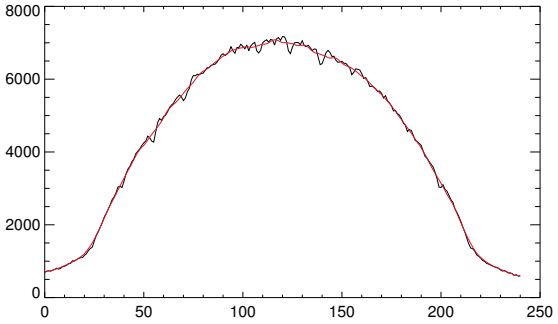
(a) The middle chord passes through the center of the sun



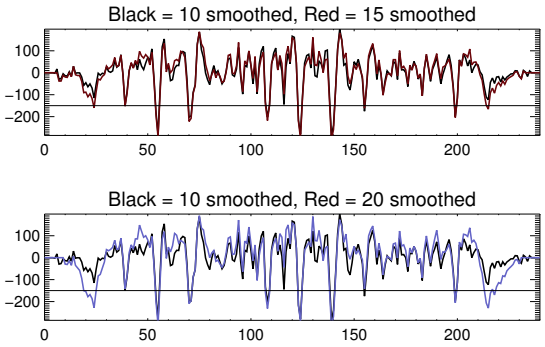
(b) The indices in red are the pixels we apply a linear fit to see where it crosses the threshold (which is the horizontal line).

Figure 4

6 Finding 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 5

Once we have a limb-fitted center, we crop a box around the center so that the sun sits comfortable inside without being clipped at the edges. We then find the sum of the image in 1D; i.e., summing the rows/columns. Certain rows/columns will have a dip in their sum (see black line in Figure 5a) from a fiducial. We smooth the sum (see red line in Figure 5a) and then subtract the two so that we have peaks at rows/columns of fiducials. (See Figure 5b) Since we don't know which pairs of coordinates go with each other, we look at all possibilities.

For each position below a certain threshold (see Figure 5b) a row/column is returned. Once we have an array of possible fiducial row and column positions, we look at each pair of coordinates to check if it is a fiducial. Our first check is to make sure the pixel value at the fiducial candidate is sufficiently dim. Many pairs end up on the solar disk but not on a fiducial. Then, we look at the 1D sum of a cropped region around a fiducial candidate. If there is a clear, whole fiducial in the cropped region, there should

be dips in the 1D sum. We smooth the 1D sum and subtract it from itself to find peaks. If these peaks are above a certain threshold (100 in Figure 7) in each axis, then the fiducial candidate is considered a *real* fiducial. Figure 6 illustrates what each fiducial candidate looks like for a certain sun.

6.1 How Robust is Fiducial Finding?

We wanted to see how our program fared when working on our smaller (449x320) original image.With the same approach, we wanted to see if the 1D sum method would return enough fiducials to accurately orient our image. For reference, the original starting image is shown in Figure 8a.

There was a problem with the lack of pixels which resulted in the `array - smooth(array,n_smooth)` plot to have peaks in the wrong places. To address this, we set an arbitrary threshold at 1000 and only smoothed pixels with higher values. Illustrated in Figure 9, this method has the potential to ignore fiducials on the very limbs of the sun, but if that is the case, then the first set of 1D sums across the entire sun would likely not pick up the faint dip in the 1D sum and would thus not even think to look there.

Figure 8b shows how our code fares when finding the fiducials of an image that is significantly smaller than our Albert-approved image.

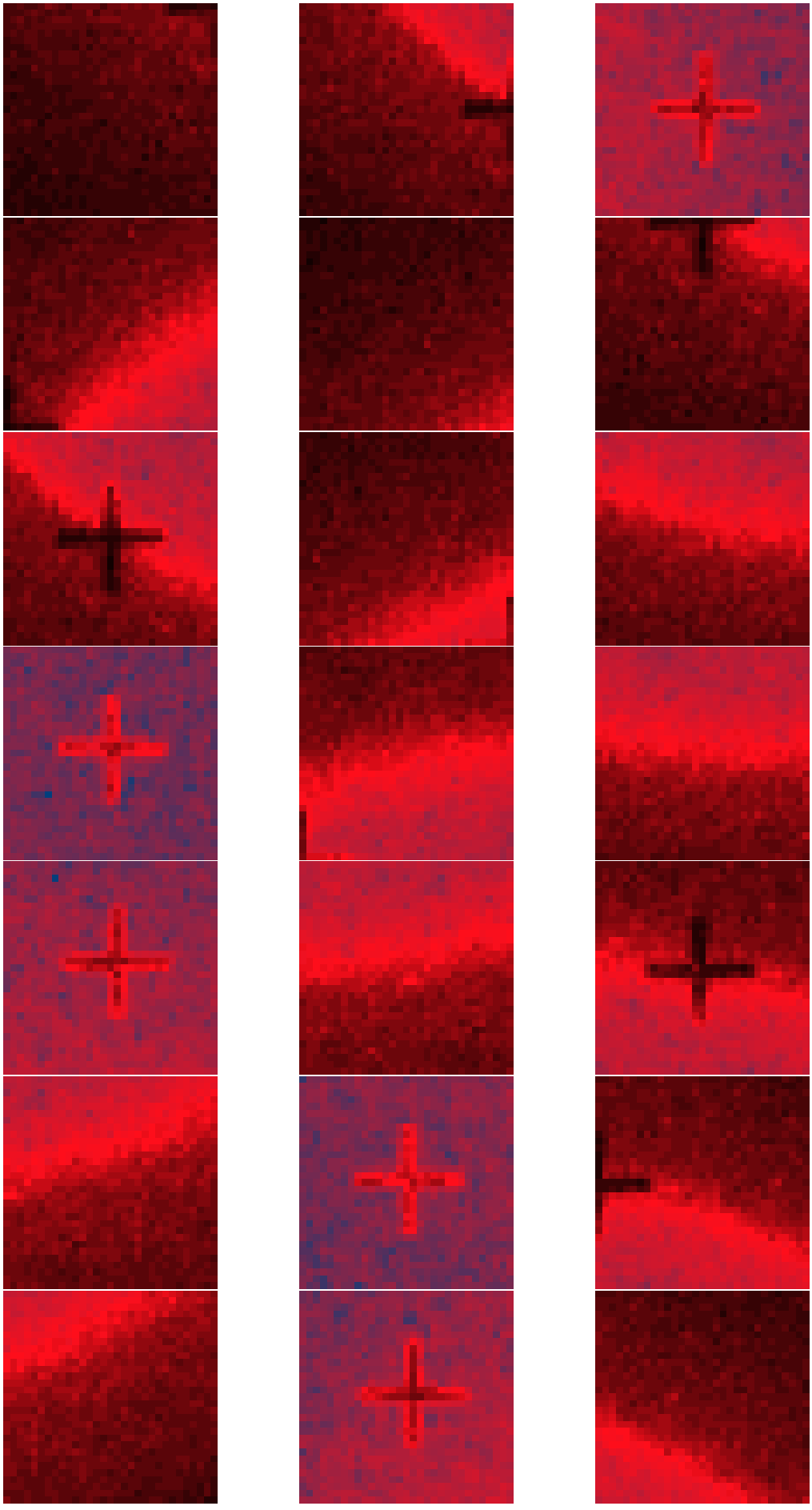


Figure 6 Each possible fiducial candidate

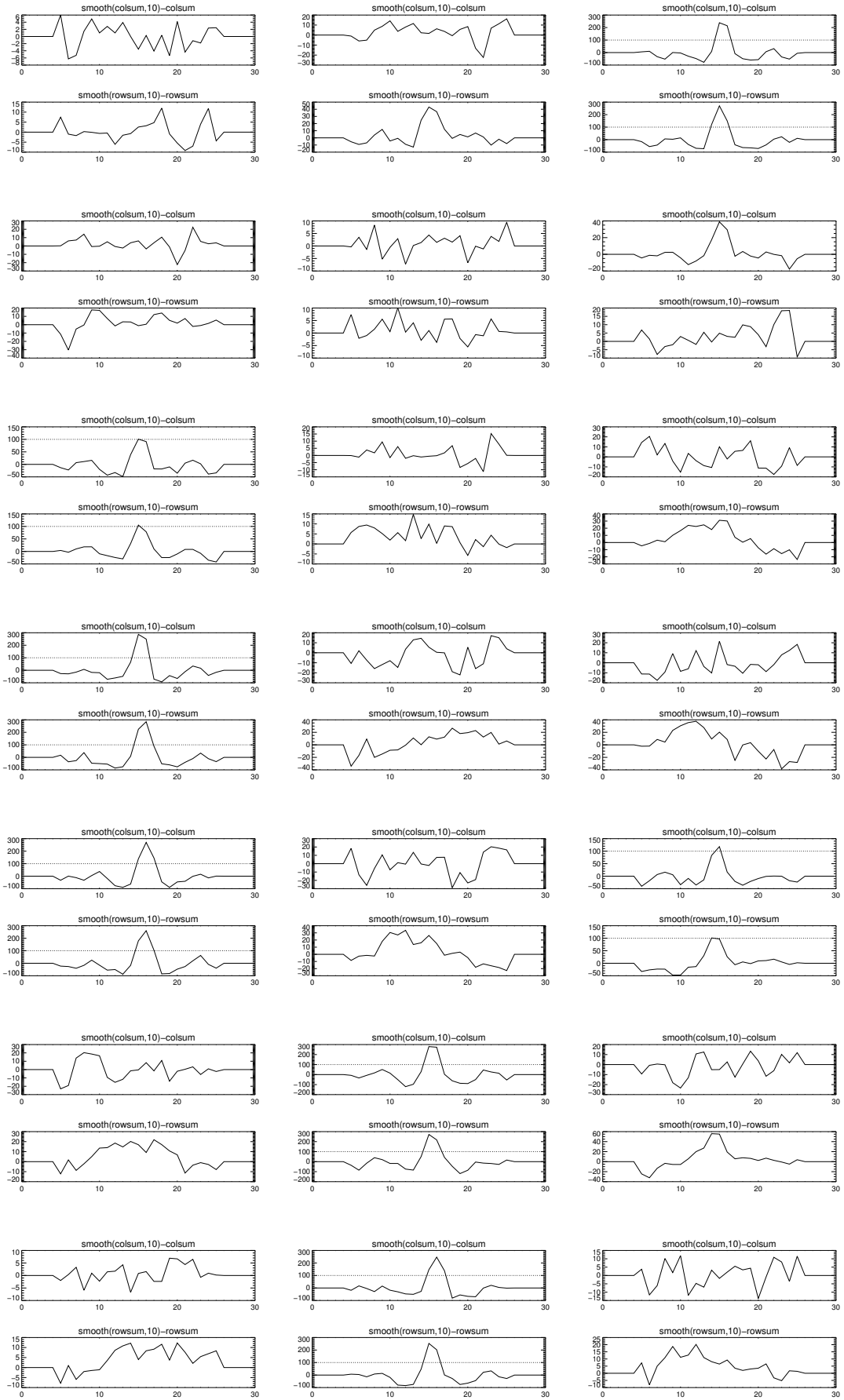
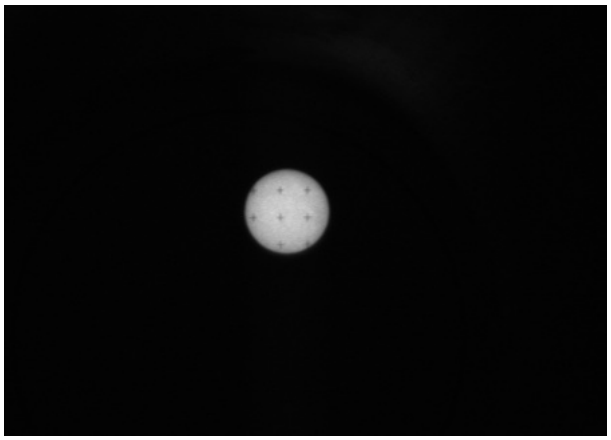
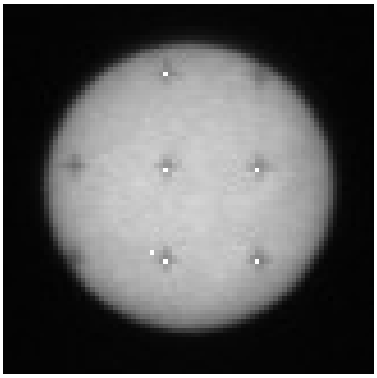


Figure 7 The position of the plots above corresponds to the same regions in Figure 6. The horizontal line is at 100; if there are any elements of the array above 100 for both a column 1D sum and a row 1D sum, then the cropped area is considered to have a fiducial in it. A parabolic fit is applied to 3 consecutive pixels with the center pixel at the peak of the array.

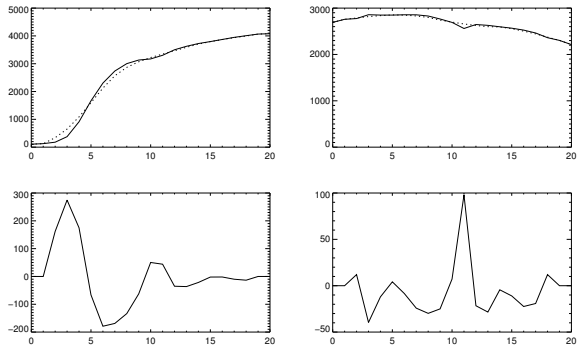


(a)

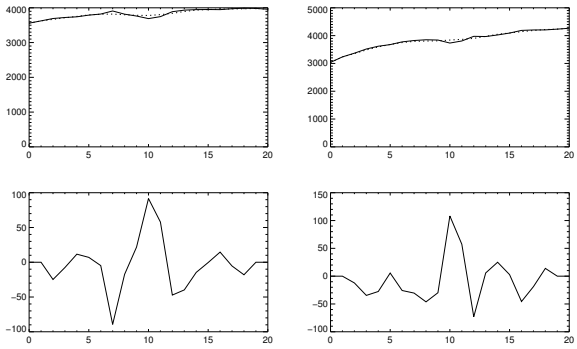


(b) Fiducial positions returned by 1D sum method on smaller, blurrier image. Ignore the bright pixel next to the bottom left fiducial.

Figure 8



(a) In the top set of plots, the solid line is the 1D sum, the dotted line is the 1D sum smoothed by 5 pixels. In the bottom set of plots, the difference of the 1D sum and the smoothed 1D sum is shown. Notice the small dip at index 10 which is dominated by the smoothing artifact of the limb at indices 0 to 5.



(b) The 1D sum is thresholded to be only above 1000 and the process is repeated, this time the difference in the two arrays returns a clear peak.

Figure 9