I Don't Care How Many Suns There Are

Jeren Suzuki

Last Edited August 27, 2013

Contents

1 Introdution

Now that we've got the *basic* framework of finding the centers of 3 suns, let's make the code flexible enough to handle each scenario:

1	2	3
1		3
1	2	
	2	3
1		
	2	
		3

Where each row corresponds to a possibility of which regions will be in the image. This table already accounts for partial suns which we do not find centers of.

2 Flexibilizing Our Code

The first step is to check if there are any partial suns, and if so, which. Once we have isolated which regions are center-friendly, we pass the region id(s) into the remainder of the program. For example, for our current setup, we'd pass reg1, reg2, and reg3 as parameters in our program. The way we identify regions is slightly worrisome to me; as of now we look at the pixel values of a region and split the image into three sections. If the max of a region is identified to fall within a specific section, it is assigned a region number accordingly. I'm not sure how robust this decision-making process is.

3 Dealing with $<$ 3 suns

How do you tell the difference between 2 suns where one is $\approx 50\%$ lower in intensity than the other? It is either a pair of region 1 and 2 suns or a pair of region 2 and 3 suns. Furthermore, how does one tell the difference from a region 1 and 2 sun that is somehow obscured to 50% brightness and a region 2 and 3 sun at normal brightness? Unless there is a way to know whether or not something is artificially making the suns dimmer, it's hard. However, if we know that and we know the approximate thresholding, we can make guesses on which region is which.

For example, if we have 2 regions, 1 is 25% the brightness of the other, we know that they are regions 1 and 3. If we only see 1 sun however, we must guess what the maximum should be and see at what percentage the max of the sun is (is it 25%? 50%?). This seems inefficient and should be improved.

Nicole suggests instead of using a foreknown number of suns we find the maxima of the 2nd deriv of the sorted array until we find no well-defined peaks. The problem with this approach is that we must define a "well defined peak". Figure 1a features the peaks of an image with the 100% and 50% sun which we try to take maximum of. Figure 1b show that if we set some arbitrary thresholds, we can get the results we want. Now, how will those threshold hold up in different scenarios?

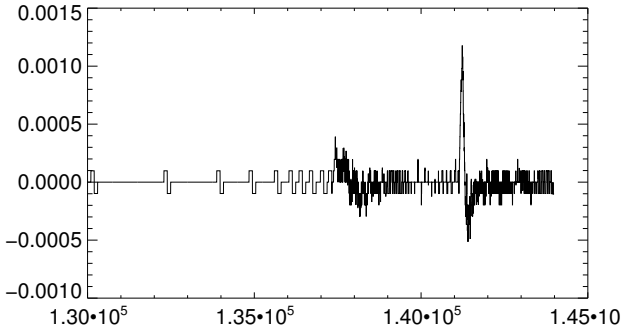
If we look at the 50% and 25% brightness sun, we see in Figure 1d that the same threshold which worked quite well in the first set of plots doesn't quite work well for others. The main problem is that we can find the maxima over and over, but we need to know when to stop. We can use `countsuns()` to return the number of suns we have. Now the problem is that we don't know which peaks are for which suns. We cannot assume the highest peak is always the 100% sun because there may be a case where only the 50% and 25% suns are available for centering.

Still working on this part.

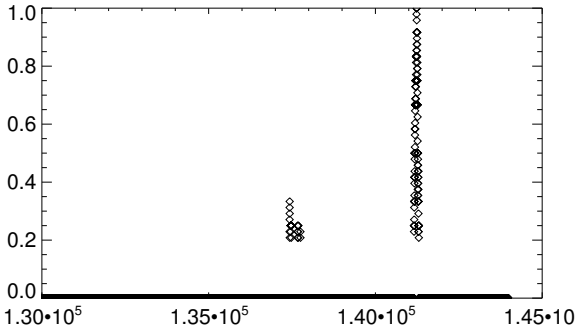
4 Partial Suns

Another big problem with identifying suns is how to deal with partial suns. We want to be able to take an image with 2 whole suns and 1 partial sun, eliminate the partial sun, and take the centers of the remaining regions. We accomplish this by:

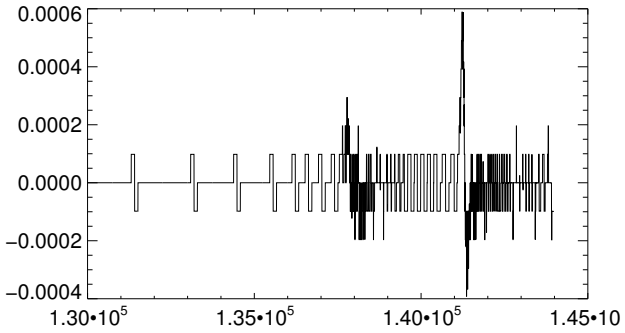
1. Reading in our image
2. Scanning the border a certain length in to check for suns



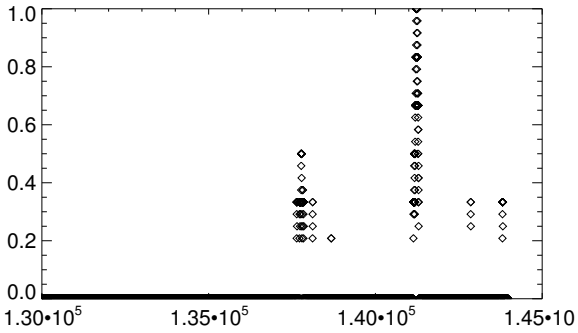
(a) 2nd deriv of sorted array of 100% and 50% suns



(b) Scaled positive values of 2nd deriv from 0 to 1, then thresholded to above .2



(c) Now looking at the 50% and 25% sun



(d) same as above

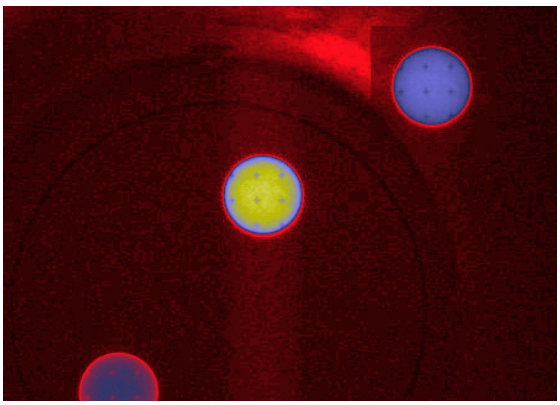
Figure 1

3. If any pixels are found, identify the partial sun as a mask and find the center of it
4. Using the center position, zero-out a box of a certain length
5. Return the fixed image with the partial sun removed

Figure 2 shows the starting and ending product of the process to fix an image. The benefit of this approach (zero out) is the simplicity and speed. The other options were to operate on the starting image but ignore the partial area or to recrop the rest of the image.

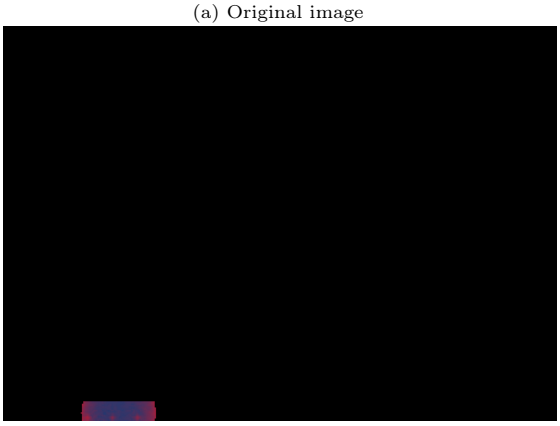
This workflow has been tested and works with the following configurations:

1. R1 & R2
2. R2 & R3
3. R1 & R3
4. R1 & R2 & R3
5. R1 & partial R2
6. R1 & partial R3
7. R1 & partial R2 & partial R3
8. partial R1 & R2
9. partial R1 & R3
10. R1 & partial R2 & R3
11. partial R1 & partial R2 & R3
12. partial R1 & R2 & partial R3

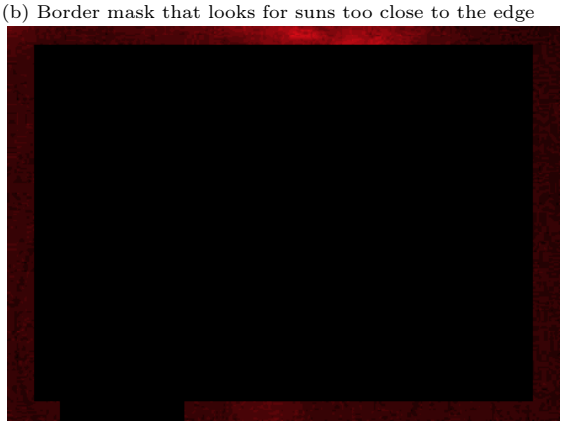


(a) Original image

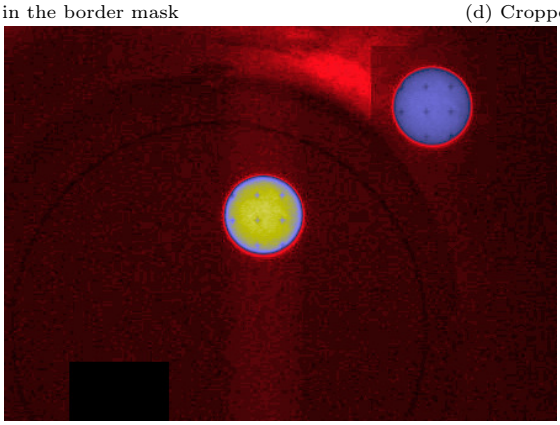
(b) Border mask that looks for suns too close to the edge



(c) Pixels above a threshold in the border mask



(d) Cropped out the partial sun



(e) Whole image with the partial sun cropped out

Figure 2: The process we use to remove partial suns and only identify the whole suns

13. partial R1 & R2 & R3
14. R2 & partial R3

Perhaps it's a good idea to get an idea of what kind of situations can arise on the craft. Will all of these occur?