Summary of ‘main.py’

By Kade Kelsch, written on 8/11/2021

The algorithm that we have currently does the following things:

1) It takes an input image
2) It extracts the RGBG2 color channels from the image
3) It saves a grayscale image of each color channel
4) It divides the image into subimages sized 170x142 (widthxheight)
    a) This specific size is because the code for manually labeling images currently uses this size
5) It passes these subimages through two filters
    a) Filter one runs through every pixel in the subimage, and at each position it determines which color channel has the highest value. Then it compares this value to that color channel’s mean and standard deviation within the subimage. A pixel is labeled as suspicious if the value has a z-score of greater than 1, and if the second highest channel value at the same position is less than sixty percent of the highest channel value.
    b) Filter two uses the pixels labeled as suspicious from filter one. At each of the suspicious pixels, it examines the 5x5 window surrounding the pixel. A pixel is labeled as suspicious if it has a z-score greater than 1.5 - where the mean and standard deviation are calculated from the 5x5 examination window of only the color channel containing the maximum value.
6) It passes the results of the second filter into a function that removes all of the suspicious pixels in each color channel image from their corresponding color images, and returns four color arrays containing the results. The suspicious pixels are removed by using a 3x3 median filter.
7) It saves a grayscale image of each of the altered color channel images.

To collect useful information on pixels that we know are noise, Dr. Pelz and our two highschool interns, Atharva Shaligram and Ethan Savage, manually labeled pixels in the top 510x284 section of certain images (this is the top two rows of 3 subimages in the upper left corner of the image). They labeled noise pixels as 'severe', 'moderate', and 'slight', according to how severe the cosmic ray damage appeared to be.

The images we decided were worth manually labeling were the following, with the corresponding metadata:

Image	F-stop	Exposure Time	ISO	Date Taken
iss031e152194	f/1.4	1 sec	12800	6/25/2021 6:16 AM
iss040e042000	f/2.8	1/8000 sec	320	7/6/2014 9:03 AM
iss041e008803	f/1.4	0.5 sec	12800	9/13/2014 3:20 PM
iss041e009489	f/1.4	0.5 sec	12800	9/13/2014 3:32 PM
iss041e009492	f/1.4	0.5 sec	12800	9/13/2014 3:32 PM
iss042e178749	f/1.4	¼ sec	100000	1/24/2015 10:02 PM
iss042e315412	f/13.0	1/1000 sec	2500	3/8/2015 9:05 AM

As we were looking at the data of the pixels that were manually identified, we decided that pixels manually labeled as being only 'slight' noise were not the priority. Our goal for the algorithm was for it to have a high hit rate, even if that also meant a high false alarm rate. According to that, below is a table illustrating the hit rate and false alarm rate of the algorithm, compared to the manually labelled data. Note that all data considered is only within the upper left 510x284 sized section of the images (this is a total of 144840 pixels).

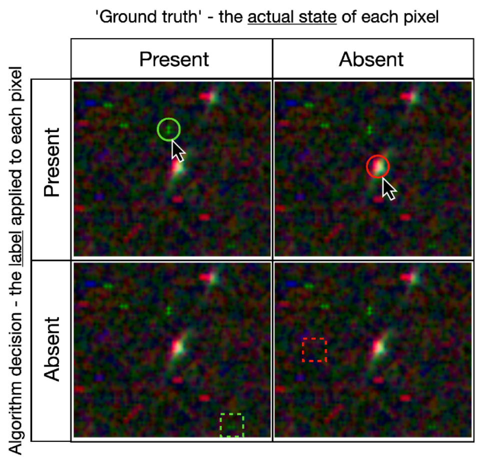
image	Number Labeled by Algorithm	Number Manually Labeled	Hit Rate	Total Number Labeled by Algorithm	False Alarm Rate	% correctly rejected
iss031e152194	57	57	100%	2877	1.9%	98.1%
iss040e042000	0	2	0%	537	0.37%	99.63%
iss041e008803	153	154	99.4%	4377	2.9%	97.1%
iss041e009489	162	162	100%	3328	2.1%	97.9%
iss041e009492	195	198	98.5%	3126	2.0%	98%
iss042e315412	28	41	68.3%	213	0.12%	99.88%
iss042e178749	71	91	78.0%	1651	1.1%	98.9%

The *Hit Rate* and *False Alarm Rate* are defined as:

Laying these two (the *ground truth* and the *label*) out as a table shows that there are four possibilities for every pixel:

'Ground truth' - the actual state of each pixel

Algorithm decision - the label applied to each pixel	Present	Absent
Present	The pixel is a CRD pixel (*present*), and the algorithm labeled it as a CRD pixel (*present*) present; present "Hit"	The pixel is *not* a CRD pixel (*absent*), but the algorithm labeled it as a CRD pixel (*present*) absent; present "False Alarm"
Absent	The pixel is a CRD pixel (*present*), but the algorithm labeled it as *not* a CRD pixel (*absent*) present; absent "Miss"	The pixel is not a CRD pixel (*absent*), and the algorithm labeled it as not a CRD pixel (*absent*) absent; absent "Correct Rejection"


'Ground truth' - the actual state of each pixel
Present
Absent
Algorithm decision - the label applied to each pixel
Present
Absent

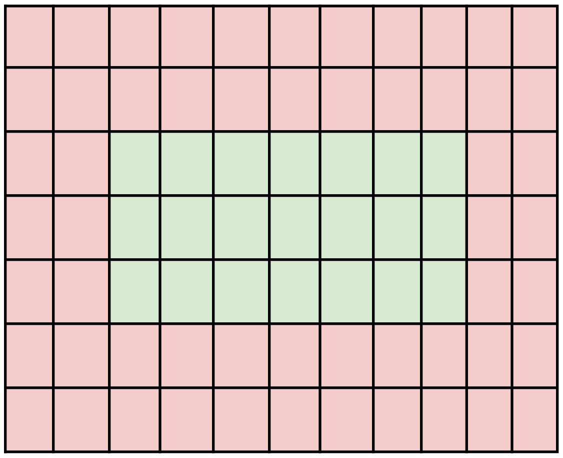
So the False Alarms should be computed as (for the case of iss041e008803):

$$FA = \frac{Number\ of\ pixels\ that\ are\ incorrectly\ labeled\ as\ CRD}{Number\ of\ pixels\ that\ are\ not\ CRD} = \frac{4377 - 153}{(510 \times 284) - 154} = 0.029\ (2.9\%)$$

In summary, the algorithm at its current state can correctly reject non - CRD pixels fairly reliably. It can also correctly identify CRD pixels at a fairly high rate of success, especially in images with longer exposure times. All pixels that the algorithm missed that were manually identified as noise were labeled as either 'moderate noise' or 'slight noise'.

Notes on the main.py:

- Because the subimage size (170x142) is manually set, not all of a full image makes it into the subimage. As in, there is a number of columns along the right side that are left out of all filtering, and a number of rows along the bottom are left out.
    - This can be fixed by making the subimage size dependent on the dimensions of the full image.
- Because the second filter uses an examination window sized 5x5, any pixels within 2 pixels of the edge of a subimage cannot be considered, because they will not have a 'full' examination window. For example, if the grid below was a subimage, red squares would be pixels that don't have a full examination window and green squares are ones that do.



    - This might be able to be fixed by applying a mirror filter, reflecting the red pixels outside of the subimage, making the red pixels accessible.
- The second filter, secondPass, can't run unless firstPass is run first. This is because it uses the data output by firstPass directly.
- Right now we have an examination window size of 5x5 for the second filter, because I found that it caught more noise compared to a 15x15 size. This is able to be easily changed if a different examination window size is desired.
- The csv files saved from firstPass and secondPass have extra data saved in them from the suspicious pixels. FirstPass also saves the mean and standard deviation of each of the color channels of that sub image, and the mean and standard deviation from the four color channel values of the suspicious pixel. SecondPass also saved the mean and standard deviation of the color channel that contains the max at the suspicious pixel, within that color channel, as well as the zscore of the max, the median and IQR of the channel that contains the max, and the MZ-score of that suspicious pixel. Much of this data is saved because it was thought it could be used to come up with a better filter, to get rid of false positives.
- In secondPass - When the IQR is 0, the MZ-score is saved as 'inf'. If we plan on using the MZ-score (modified z-score based on the median) in a filter at some point, this will need to be addressed somehow. Below is a table of the ranges of MZ-scores that existed from the manually labeled pixels of the images.

image	Minimum MZ-score	Maximum MZ-score
iss031e152194	0.67	38
iss040e042000	NA	NA
iss041e008803	0.76	28
iss041e009489	0.9	43
iss041e009492	0.89	34
iss042e315412	3.5	117
iss042e178749	1.4	552

Image iss040e042000 does not have any data because the algorithm did not detect any of the manually labeled noise pixels.

The calculation for the MZ-score is:

0.6745*((max - median)/IQR

I (Kade Kelsch) personally do not think that the MZ-score would be useful as a filter, given the range of values it takes on for the manually labelled noise.

- The median filter currently used to replace the suspicious pixels in each color channel uses a 3x3 examination window to select the median from. It only replaces suspicious pixels in the channel that contains the suspected cosmic ray damage.
- In order to save grayscale images of the color channels, both before altering and after, each channel image has to be converted to 8-bit. Right now the code is written to convert from 16-bit to 8-bit, but if images of a different bit size are going to be processed by the code, this will need to be altered.
- When we display the grayscale images of the colorchannels, we use cv.equalizeHist to use histogram equalizing to increase contrast of the images - this may not be the best method of increasing the contrast on all images. It works very well for showing the ‘before’ and ‘after’ images for iss041e008803, but does not work so well for iss031e152194 or iss042e315412.

Ideas for the Future:

- Many of the pixels that were manually labelled that the algorithm did not label (the misses) were not labeled as suspicious in firstPass, usually because they had a z-score of less than one. This was usually due to a large mean and/or variation - the mean and variation are based on the subimage (142x170). This could be fixed by lowering the minimum z-score, or possibly making the subimages smaller (Jeff suggests trying this first). Another idea is to place a condition filter based on how large the variation - if the variation is above some value, try breaking up the subimage.
- An idea for another filter would be a grayscale filter. Instead of examining each of the color channels individually, we would examine the pixel in a grayscale version of the rendered image. This would be specifically useful in filtering out stars suffering from chromatic aberration from noise.
- Although I do not think that the MZ-score will be useful in making another filter, I think using the standard deviation of the color channel that contains the maximum value in the 5x5 examination window might be more useful. However, between images the lower threshold for the standard deviation changes. I think it might be able to be calculated based on the metadata of the image.

image	Minimum Standard Deviation
iss031e152194	8
iss040e042000	NA
iss041e008803	30.8
iss041e009489	27
iss041e009492	30.5
iss042e315412	2.8
iss042e178749	4.9

- The general form of the image data varies depending on some of the camera settings. For instance, there is more background noise when there is a longer exposure time and a high ISO. This affects some of the statistics within the subimages and examination windows. If a filter's parameters could be adjusted based on some of the metadata of the images, that would likely help in coming up with more specific filters to filter out false positives. I don't have any ideas on how to do this or what kind of filter, as this is a recent idea. (Jeff notes: this is probably going to be very useful)

If there are any questions that you need me to answer, I can be reached at akkatek0@gmail.com. The github that contains my most updated version of this code is at https://github.com/akkadek/SvN.