Draft version February 21, 2021

Typeset using LaTeX **twocolumn** style in AASTeX63

Radial Normalization of Solar Images

Chris R. Gilly[1, 2] and Steven R. Cranmer[1, 2]

[1] *Laboratory for Atmospheric and Space Physics*
[2] *Astrophysical and Planetary Sciences Department, University of Colorado Boulder*

ABSTRACT

In this work we show a straightforward method to make solar images have a constant dynamic range with radius. Pixels are sorted by radial distance, then the 95th and 5th percentiles are found. These noisy curves are then fit with an polynomial function to provide a smooth scaling. Intensities are normalized between zero and one, and outliers are treated separately. A vignette is then applied to the image to truncate the noise floor.

This reduction is being performed in perpetuity on an AWS t2.micro instance, with NRT imagery hosted at The Sun Now

The reduction engine is written in python and can be found on GitHub

1. INTRODUCTION AND MOTIVATION

Because the intensity of the solar corona drops off rapidly with heliocentric altitude, it is very difficult to take images of both the sun's disk and the overlaying corona with the same instrument. There are many methods in the literature that can be used to expose details in the raw science images: taking the square root or logarithm of the entire array, multiplying each pixel by its radius r^{NUM}, Multi-scale Gaussian Normalization (**?**), and Noise Gating(**?**), to name a few.

In this work, we introduce a novel algorithm to normalize images of the sun's corona using percentiles.

2. METHOD

This section will describe the algorithm used in the python program located on GitHub.

An image I (either a fits file, file path, or a np.ndarray) is passed into the Modify class (see Appendix A for usage instructions). The reduction pipeline operates as follows:

1. Load the image I. Add min(I) to I, setting the minimum intensity to 0.
2. Compute a radius array R so that each pixel has a known $r = \sqrt{x^2 + y^2}$.
3. Flatten I and bin the intensities by radius. This produces a cloud of intensity values at each bin height. Intensities are shown as black points in Figure 1(a).
4. Compute an upper and lower percentile of the intensities in each bin as function of radius. Shown as colored curves in Figure 1(a)). In this work, the high cutoff used was 75th and low was 2nd.
5. Use a Savgol filter to smooth the percentile curves, treating the regions independently.
6. Fit a polynomial to the on-disk curve to help mitigate over-bright regions.
7. Concatenate the regions and Savgol smooth them again.
8. Normalize the input image using these curves - standard formula
9. Handle outliers like NANs, hot, and cold pixels
10. Vignette the image, truncating the image at a given radius

Figure 1 (a) shows the intensity values at each height in black, along with the 75th and 2nd percentile curves. Figure 1 (b) shows the result after normalizing the intensities to these percentile curves.

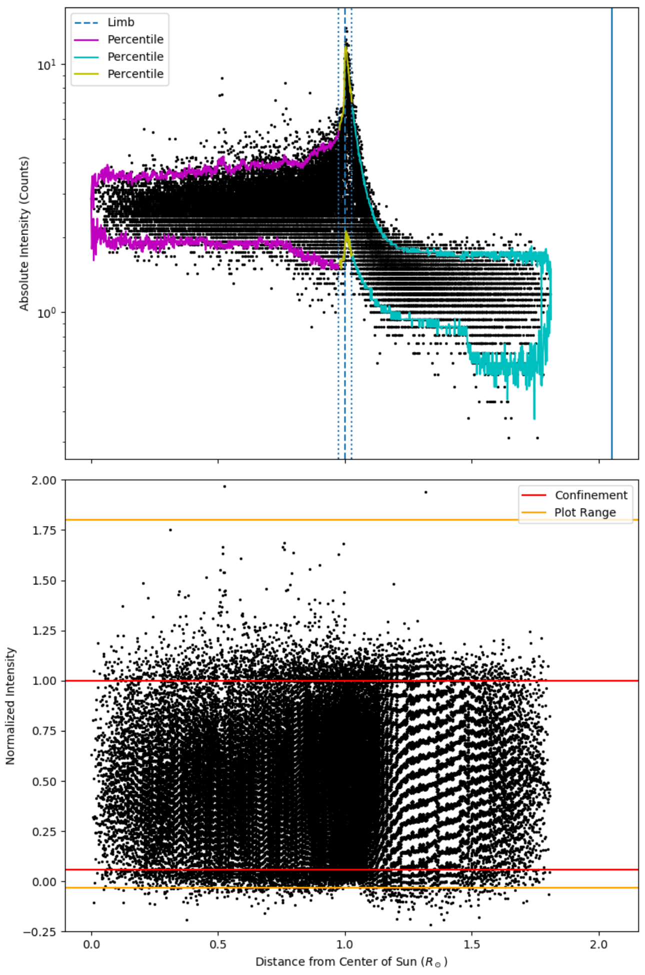

Figure 1. Pixel Intensity vs Radius

Figure 2 compares a processed image beside the original.

3. FUTURE WORK

There is a review paper about solar coronagraphy in preperation which will compare and contrast the different methods in the literature.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Mike Hahn, Ben Boe, and Sam van Kooten for many valuable discussions. This work was supported by the National Aeronautics and Space Administration (NASA) under grants NNX15AW33G and NNX16AG87G, and by the National Science Foundation (NSF) under grants 1540094 and 1613207. Thanks also to Juri Toomre's Laboratory for Computational Dynamics for providing us with computing time.

Software: NASA's Astrophysics Data System (ADS), Python version 3.6.5 (?), numpy (?), scipy (?), scikit-image (?), MPI4py (?), matplotlib (?), and CHIANTI v8 (?)

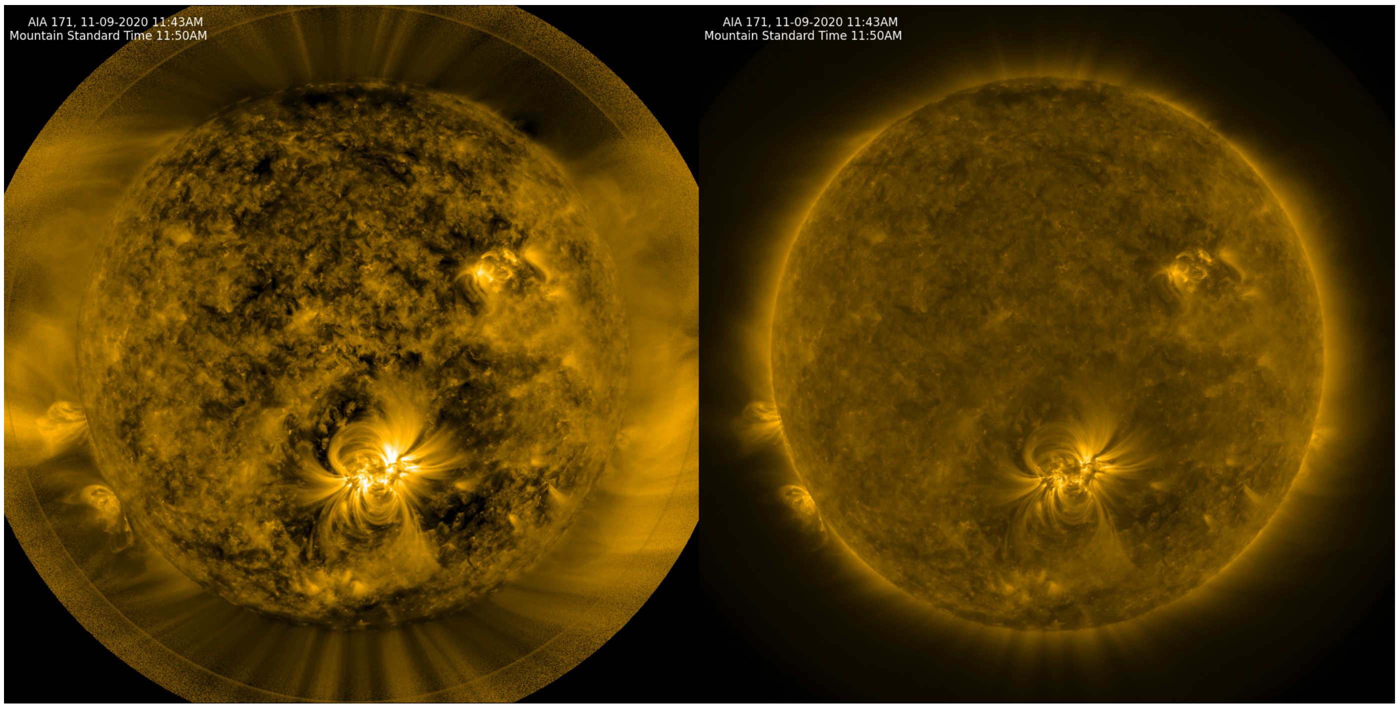
AIA 171, 11-09-2020 11:43AM
Mountain Standard Time 11:50AM
AIA 171, 11-09-2020 11:43AM
Mountain Standard Time 11:50AM

Figure 2. Normalized Image on Left, Original on Right

APPENDIX

A. GETTING STARTED

To get a local copy up and running follow these simple example steps.
Prerequisites: You need python 3, and Conda is probably your best bet.

A.1. *Installation*

1. Clone the repo

```
git clone https://github.com/GillySpace27/sunback_engine.git
```

2. Install required packages

```
pip install -r requirements.txt
```

3. Confirm it works (tests will run)

```
python sunback_filt.py
```

A.2. *Usage Examples*

To perform the reduction using a fits file, you can provide the path to the file to the Modify Class:

```
from sunback_filt import Modify
input_path = "/data/171_MR.fits"
reduced = Modify(input_path)
```

One can also perform the reduction directly on any numpy NDArray:

```
from sunback_filt import Modify, load_file
input_path = "/data/171_MR.fits"
input_file = load_file(input_path)
reduced = Modify(input_file)
```

Defaults for Modify (Top Level Class):

```
Modify(data=None, image_data=None, orig=False, show=False, verb=False)
```

with full_name, save_path, time_string, shape = self.image_data