

SUN-SLIC: A Training-Free Superpixel Method for Coronal Hole Segmentation in SDO/AIA 193 Å Images

Gábor Funk

October 2026

Abstract

Coronal holes, the dark, low-density regions of the solar corona that source fast solar wind, are commonly segmented in extreme ultraviolet images using convolutional neural networks or hand-set intensity thresholds. Neural approaches require labeled training data and substantial compute, while thresholds are sensitive to image conditions. I present SUN-SLIC, a modified SLIC superpixel algorithm adapted to SDO/AIA 193 Å images, which segments coronal holes without any training. Superpixels are clustered by weighted k-means on consistently found texture features, restricted to a central disk mask which were defined by us, and merged across small gaps by morphological closing, after which separate holes are counted by connected-component analysis. I evaluated the method on images from 2026 October and January, obtaining a mean IoU of 0.55 and detecting 95% of catalogued holes. Parameters were chosen on a separate tuning set, and a sensitivity analysis shows the result is stable over the dataset I used for validation. The method runs in avg. 0.28 seconds per image on a CPU, and its limitations, such as confusion with filaments near the limb, are discussed.

Contents

1	Definitions	1
2	SUN-SLIC	2
2.1	Pseudo code	3
3	Example output	4
4	Runtime complexity	5

1 Definitions

I know that the Sun’s active parts are very bright in SDO/AIA 193 Å images while the coronal holes are dark-reddish with a gradient texture indicating the center of them.

The algorithm I propose is able to find large and small coronal holes, their relative size to their image and other active parts of the Sun or when fine-tuned, any star with the correct parameters.

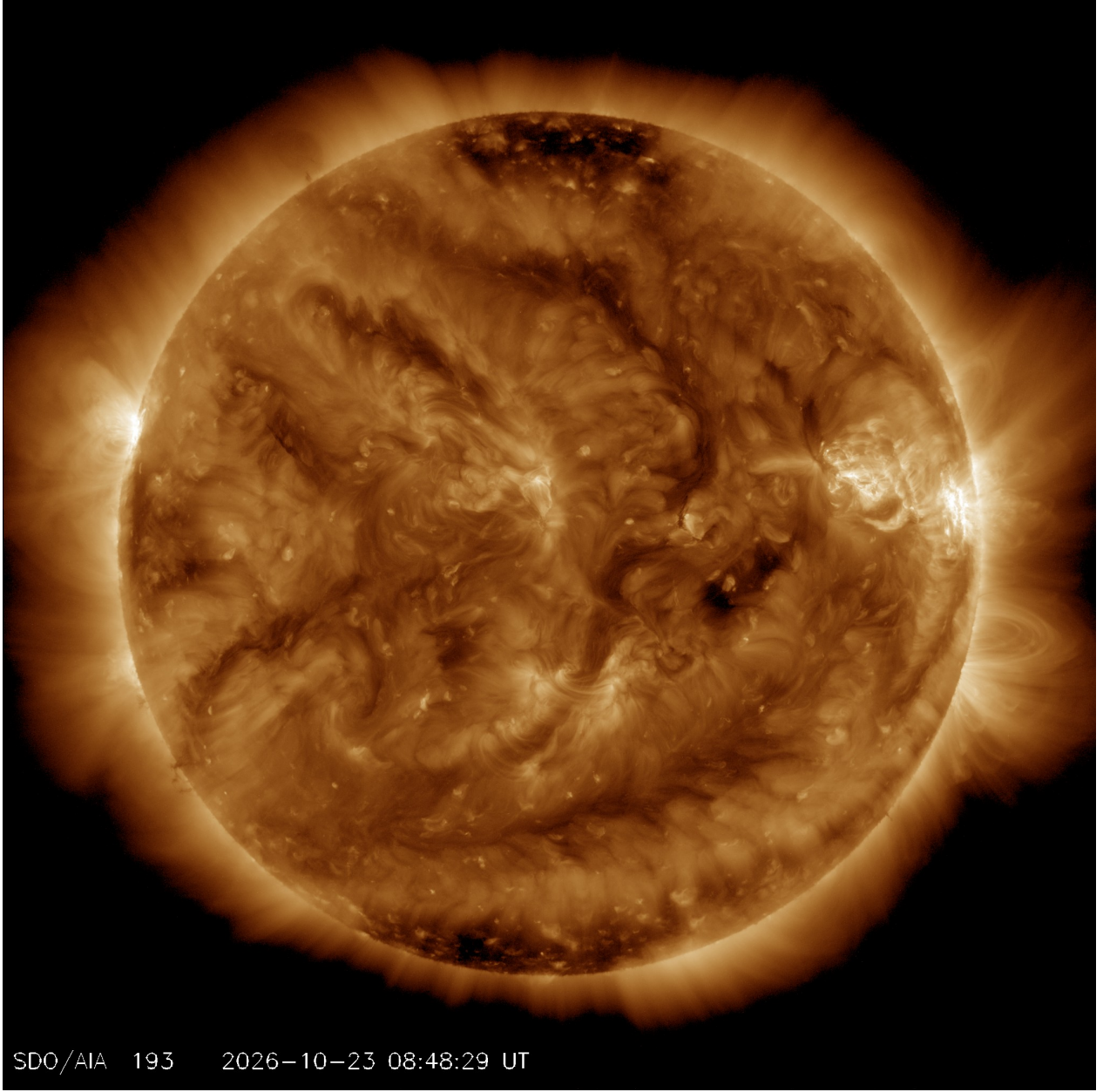


Figure 1: An example image from the dataset. Smaller coronal holes are visible in it.

2 SUN-SLIC

The algorithm is a modified SLIC algorithm with Floodfill and LLoyd’s algorithm which is non-deterministic but can be replaced to be deterministic. The output will be 2 sliced images, one for the active part labels and one for the coronal hole labels. The algorithm will additionally output the detected coronal hole amount and their size on the image in pixels. This must be converted to other units if needed.

2.1 Pseudo code

An implementation can be found at https://afghangoat.hu/SLIC/sun_slic.html in JavaScript language as well as an online demo.

A *MAXD* variable will control how much the image will get scaled down. This algorithm works for lower scale images due to gradient packing.

```
INPUT the image
Scale the image down to have MAXD as its larger sidelength
Convert the pixels of the image to lab color for SLIC
Generate labels with SLIC
CALL enforceConnectivity ON previously calculated data
    for each label group:
        check the neighboring labels
        if the group's size is smaller than C pixels (C is a tuned variable)
            merge the group into a neighboring one
        else:
            keep it
END enforceConnectivity
CALL superpixelFeatures ON previously calculated data
    for each superpixel
        calculate:
            mean L,
            a (position),
            b (position),
            std of L (a crude texture measure,
            flat sky has almost 0, fur for example has large value),
            pixel count of superpixel
    #Superpixels become a classification vector
    return the calculated classification vectors
END superpixelFeatures
CALL K-means ON previously calculated data
    Lloyd's algorithm with R restarts and I iterations

    Look for k centers that minimize the sum of
    squared distances from points to their nearest center.

    return the K-means
END K-means
CALL extractObjects
    for each label region:
        lum = i => (px[i*4] + px[i*4+1] + px[i*4+2]) / (3.0 * 255.0)
    sort regions based on luminosity
    activeRegions = region with maximum luminosity.

    #This is needed to skip other stars and the left-bottom label
    AA = region which gets covered with:
        unit circle with center as image center,
        R=40% of downscaled image's smaller sidelength.
```

```

coronalHoleRegions = region with least luminosity in AA region

COUNT components which have more pixels than FFC with Floodfill
#The outputs are the described images and their cluster params.
return result
END extractObjects

```

FFC is the Floodfill constant. This filters out the coronal hole looking parts which are not coronal holes but indeed only light perturbations.

$$FFC = MAXD/7.85714285$$

This is a fitted heuristic value.

3 Example output

For the earlier example image, the algorithm outputs:

latest_2048_0193.jpg - brightest (active) - Lightness: 0.6344 (7300 px, 10 separate possible POIs) (sizes: 426,425,279,226,72,69,68,66,66,65)



latest_2048_0193.jpg - 2nd darkest (coronal holes) - Lightness: 0.1640 (9567 px, 12 separate possible POIs) (sizes: 120,104,88,86,82,81,81,80,64,64,63,62)



Figure 2: The 2 output images, their detection counts and sizes.

4 Runtime complexity

The naive per-component full-image scan algorithm has a runtime complexity of:

$$T_{\text{naive}} = \mathcal{O}(N * T + N + C * N + R * I * K * k * d), \quad C \geq k \quad (1)$$

where N is the pixel count, T the SLIC iterations, K the number of superpixels, k the number of k-means clusters, R the random-restarts, I the k-means iterations, d the feature dimension, C the number of connected components, and A_c the bounding-box area of component c .

References

- [1] SDO dataset
<https://sdo.gsfc.nasa.gov/data/>