;Steps followed in original “general\_full\_cadence.pro”

1. Find fits files
2. Create 2048 x 2048 mask in the shape of a disk for pixels larger than 800 (not clear if this is the size of the Sun). Fits images are originally also 2048 x 2048.
3. read in and perform the latest calibration and image correction procedures:

secchi\_prep, files(q), outhdr, img, outsize = 2048, /rotate\_on, /calimg\_off, /cubic

;output image as "img"

;outsize: 2048

;outhdr : headers

;/CALIMG\_OFF: Do not apply vignetting or flat-field calibration

;/ROTATE\_ON: the images are rotated to solar north. The difference between solar north and ecliptic north changes though out the year.

;/CUBIC (maybe, not sure) = Interpolation parameter for cubic interpolation. See the IDL documentation for POLY\_2D for more information.

1. Take natural log of the image for pixels above 0.01 (new “img”)
2. Replace pixels more than a specified pixels deviant from its neighbors with sigma\_filter (new “img”). I think border pixels are not processed (need to check this in more detail).

img = sigma\_filter(img,radius=**3**,/iterate)

Compute mean over moving box-cars using smooth, subtract center values, compute variance using smooth on deviations from mean, check where pixel deviation from mean is within variance of box, replace those pixels in smoothed image (mean) with original values, return the resulting partial mean image.

;RADIUS = alternative to specify box radius, so box\_width = 2\*radius+1

;N\_sigma = # standard deviations to define outliers, floating point, default = 3, smooths 1% of pixels.

;/ITERATE causes sigma\_filter to be applied recursively (max = 20 times) until no more pixels change.

1. Only for pixels outside circle of mask, apply sigma\_filter with larger radius (new “img”)

img(mask\_idx) = (sigma\_filter(img,radius=**13**,/iterate))(mask\_idx)

1. Smooth image 52 times with boxcar of 30 (new “mr” image). Edges are processed because kconvol adds extra mirrored pixels on the side of the image (see below)

mr = kconvol(img,30)

for t=0,50 do mr = kconvol(mr,30)

It invokes SMOOTH(mr,30).

SMOOTH(A,w):

A picture containing text

Description automatically generated

where *w* is the smoothing width and *N* is the number of elements in A

1. Apply kernel twice (new “img1”) and thrice (new “l0”).

A close up of a sign

Description automatically generatedKernel k used:

img1 = kconvol(img,k,total(k))

img1 = kconvol(img1,k,total(k))

l0 = kconvol(img1,k,total(k))

kconvol adds these extra-pixels to the sides (original image in yellow):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| A16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A14 | A04 | A04 | A14 | A24 | A34 | A44 | A54 | A64 | A64 | A54 |
| A13 | A03 | A03 | A13 | A23 | A33 | A43 | A53 | A63 | A63 | A53 |
| A12 | A02 | A02 | A12 | A22 | A32 | A42 | A52 | A62 | A62 | A52 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |

Csh

sswidl

--------------

aaa = **dist**(**2048**,**2048**)

imgsize = **size**(aaa)

aaaplot = **IMAGE**(aaa , IMAGE\_DIMENSIONS=[imgsize(**1**),imgsize(**2**)], DIMENSIONS=[**512**,**512**], MARGIN=**0**)

A close up of a logo

Description automatically generated

maskplot = IMAGE(mask , IMAGE\_DIMENSIONS=[imgsize(1),imgsize(2)], DIMENSIONS=[512,512], MARGIN=0)

Shape

Description automatically generated

result = **IMAGE\_THRESHOLD**(mask , THRESHOLD=maxmask)

maskplot = **IMAGE**(result , IMAGE\_DIMENSIONS=[imgsize(**1**),imgsize(**2**)], DIMENSIONS=[**512**,**512**], MARGIN=**0**)

**Mask: chooses outside the disk**

Icon

Description automatically generated

img = READFITS( files(q), outhdr, /NOSCALE)

imgsize = **size**(img)

implot = **IMAGE**(img , IMAGE\_DIMENSIONS=[imgsize(**1**),imgsize(**2**)], DIMENSIONS=[**512**,**512**], MARGIN=**0**)

A star in the dark

Description automatically generated

img = alog(img>0.01)

A picture containing sitting, photo, table, monitor

Description automatically generated

img = sigma\_filter(img,radius=3,/iterate)

A picture containing sitting, photo, table, apple

Description automatically generated

img(mask\_idx) = (sigma\_filter(img,radius=13,/iterate))(mask\_idx)

max =800

A picture containing sitting, photo, table, monitor

Description automatically generated

Max= 400

A picture containing sitting, table, photo, apple

Description automatically generated

A picture containing table

Description automatically generatedTable

Description automatically generated

VIII

VII

VI

V

VIII

VII

VI

V

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| A16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A14 | A04 | A04 | A14 | A24 | A34 | A44 | A54 | A64 | A64 | A54 |
| A13 | A03 | A03 | A13 | A23 | A33 | A43 | A53 | A63 | A63 | A53 |
| A12 | A02 | A02 | A12 | A22 | A32 | A42 | A52 | A62 | A62 | A52 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |

A close up of a sign

Description automatically generated

Total(k) = 11

Convolve

**kconvol**(A,k,**total**(k))

**CONVOL**(A,K,scale\_factor)

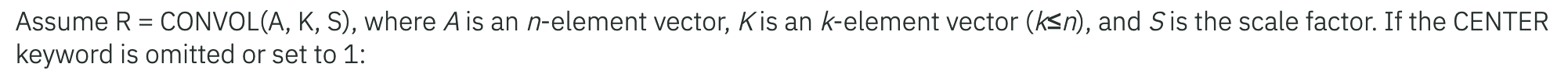
A scale factor that is divided into each resulting value.

the kernel is centered over each data point.

In the two-dimensional, zero CENTER case where *A* is an *m*-by-*n*-element array, and *K* is the *k*-by-*k* element kernel; the result *R* is an *m* by *n*-element array:

It is usually not appropriate to divide the result value by the full scale factor if portions of the kernel were not applied due to missing data. In this case, you might want to use the [NORMALIZE](https://www.harrisgeospatial.com/docs/CONVOL.html#C_854643309_1297207) keyword instead.

If NORMALIZE is set and your input array has missing data (the [INVALID](https://www.harrisgeospatial.com/docs/CONVOL.html#C_854643309_1297467) or [NAN](https://www.harrisgeospatial.com/docs/CONVOL.html#C_854643309_1028683) keywords are set), then for each result value the scale factor and bias are computed using only those kernel values that contributed to that result value. This ensures that all result values are comparable in magnitude, regardless of any missing data. Use caution when analyzing these values, as the result may be biased by having fewer points within the kernel.



Text

Description automatically generated

*S* is the scale factor.

Smooth:

**SMOOTH**(A,kernel)

A picture containing text

Description automatically generated

where *w* is the smoothing width and *N* is the number of elements in A

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| 16 | A06 | A06 | A16 | A26 | A36 | A46 | A56 | A66 | A66 | A56 |
| A15 | A05 | A05 | A15 | A25 | A30 | A45 | A55 | A65 | A65 | A55 |
| A14 | A04 | A04 | A14 | A24 | A34 | A44 | A54 | A64 | A64 | A54 |
| A13 | A03 | A03 | A13 | A23 | A33 | A43 | A53 | A63 | A63 | A53 |
| A12 | A02 | A02 | A12 | A22 | A32 | A42 | A52 | A62 | A62 | A52 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A10 | A00 | A00 | A10 | A20 | A30 | A40 | A50 | A60 | A60 | A50 |
| A11 | A01 | A01 | A11 | A21 | A31 | A41 | A51 | A61 | A61 | A51 |