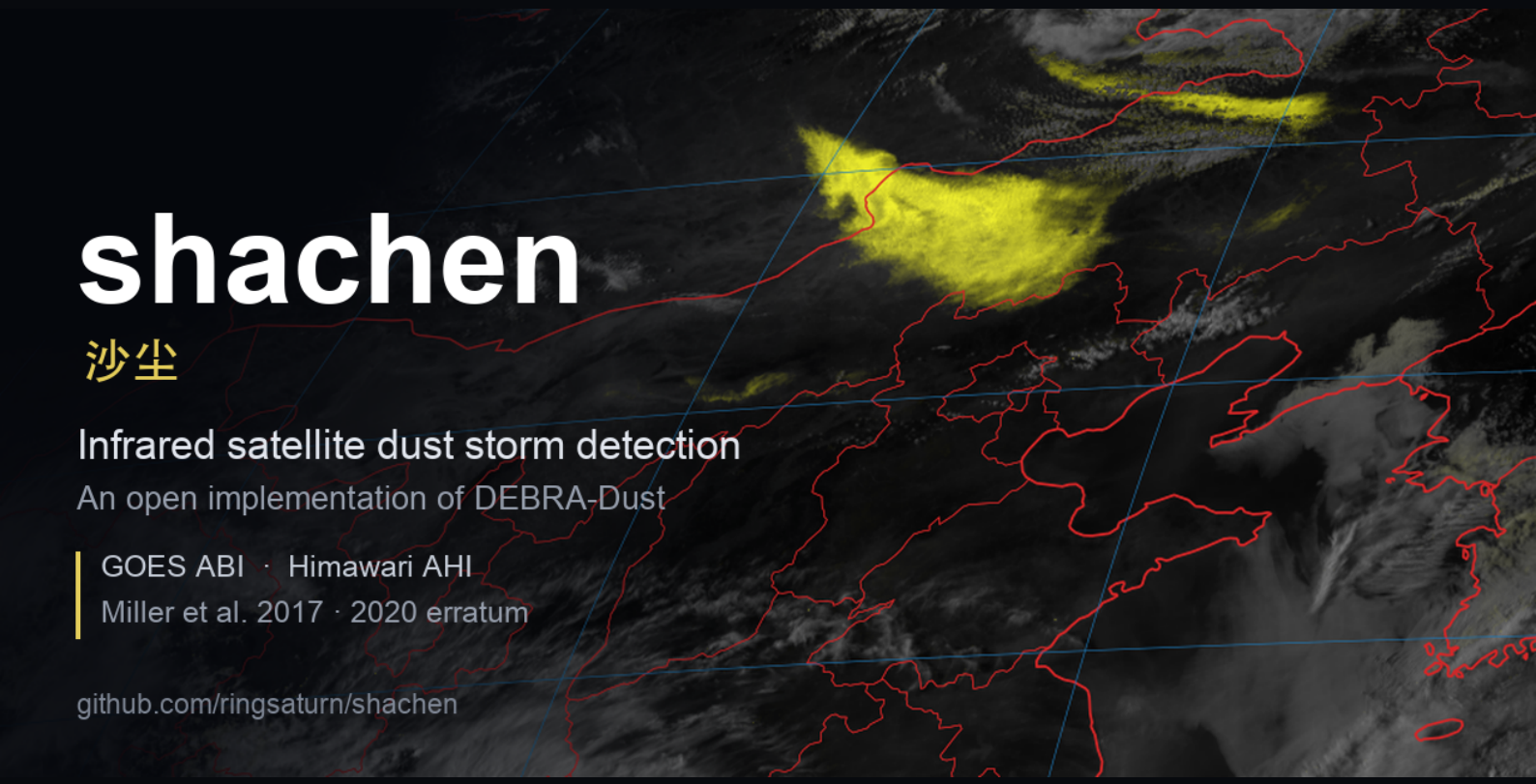


shachen

沙尘

Infrared satellite dust storm detection

An open implementation of DEBRA-Dust

GOES ABI · Himawari AHI
Miller et al. 2017 · 2020 erratum

github.com/ringsaturn/shachen

shachen — DEBRA-Dust

Release 0.2.3

Han Xiao

Aug 30, 2026

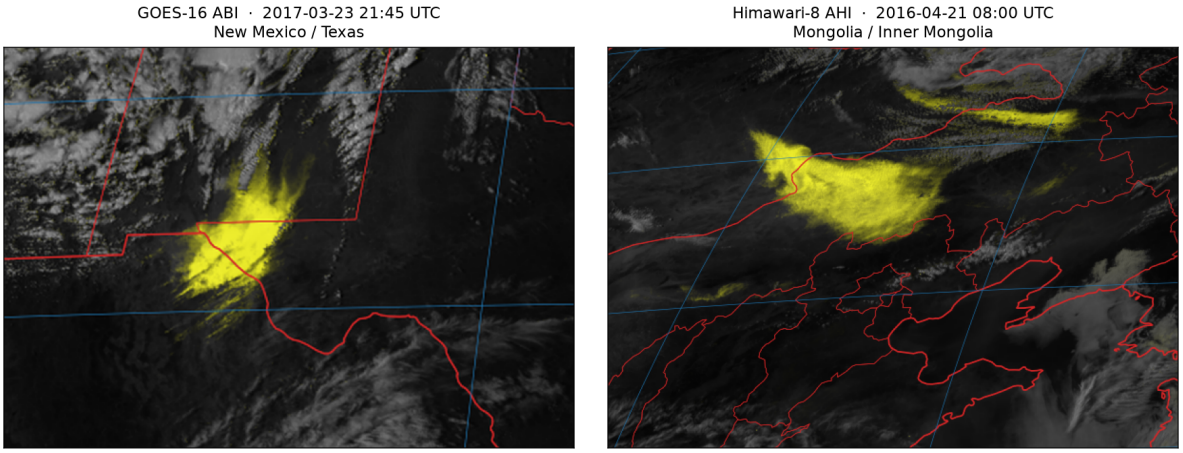
CONTENTS

Infrared satellite dust storm detection in Python — an open implementation of DEBRA-Dust, the Dynamic Enhancement with Background Reduction Algorithm (Miller et al. 2017[1]), for GOES ABI and Himawari AHI.

shachen (沙尘) is Chinese for "sand and dust". The package is a home for infrared-channel dust algorithms; DEBRA-Dust is the first one.

Equations follow the 2020 erratum

All equations follow the erratum published 26 February 2020, which supersedes the 2017 print run for Eqs. 7, 21–22 and 24–25, plus three further corrections documented in *Deviations*. Read that page before changing any constant or sign convention.



Dust is the yellow modulation; everything else stays in greyscale infrared.

[1] https://doi.org/10.1002/2017JD027365

CHAPTER

ONE

WHAT IT DOES

DEBRA turns the split-window infrared signal that is specific to mineral dust into a per-pixel confidence field, by comparing each pixel against a dynamically estimated clear-sky background rather than a fixed threshold. Over bright, emissivity-heterogeneous desert surfaces, fixed thresholds produce false alarms; the dynamic background removes that dependence.

```
io.satellite.load_scene   L1b → bt_* / refl_* on the 2 km fixed grid (satpy)
        │
        ▼
pipeline.run_debra
├─ geo.regrid_latlon      MERRA-2 / CAMEL → satellite grid
├─ solar                  per-pixel solar zenith (day / twilight / night mix)
├─ background             scheme A: CAMEL emissivity × Planck(MERRA-2 skin T)
│    or composite         scheme B: 14-day cloud-cleared same-hour composite
├─ cloudmask              Eqs. 1–12, including the dust restoral term
├─ dust_tests             DT1–DT3, Eqs. 13–15, normalised per-pixel
├─ confidence             Eqs. 16–22 → cf_comb
├─ imagery                Eqs. 23–29, CF-modulated RGB
└─ render                 georeferenced PNG with coastlines (cartopy)
```

CHAPTER

TWO

QUICK START

```
pip install shachen
```

```
import shachen

result = shachen.run_debra(scene, skin_temperature=merra_ts, emissivity=camel)
result["cf_comb"]  # combined dust confidence, 0–1
```

The *User guide* covers what `scene` must contain, the two background schemes, and how to get from a confidence field to a rendered PNG. *Equations* sets out all 29 of the paper's equations in the form this package implements them, each linked to the function that runs it, and the *API reference* documents every module.

2.1 User guide

2.1.1 Install

The core install runs the entire algorithm on fields already in memory, and pulls in no I/O or plotting stack:

```
pip install shachen
```

Optional extras:

```
pip install "shachen[satellite]"   # satpy: read and calibrate ABI/AHI L1b
pip install "shachen[data]"        # earthaccess/s3fs: fetch MERRA-2, CAMEL, L1b
pip install "shachen[render]"      # cartopy/matplotlib: georeferenced PNG
pip install "shachen[all]"         # everything, for the reproduction scripts
```

`import shachen` pulls in none of the extras; CI asserts that.

2.1.2 The scene contract

Every stage works on the sensor's 2 km fixed grid. A *scene* is an `xarray.Dataset`[2] whose variables are named by DEBRA role rather than by sensor channel, so ABI and AHI share one code path:

[2] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

Variable	Unit	Role
`refl_vis_064`	%	daytime baseline image (Eq. 23)
`refl_nir_160`	%	reserved for the daytime cloud test
`bt_swir_39`	K	night thin-cirrus test CM4 (Eq. 6)
`bt_wv_62`	K	deep-convection test CM2 (Eq. 4)
`bt_tir_86`	K	dust test DT2 (Eq. 14)
`bt_tir_104`	K	clean window reference
`bt_tir_123`	K	dirty window, RSW / DT1 (Eq. 13)

Two attributes must be present:

`scene.attrs["area"]`
: a `pyresample.geometry.AreaDefinition`[3] — the projection and grid every ancillary field is interpolated onto, and the map projection `render` draws in.

`scene.attrs["start_time"]`
: a naive-UTC `datetime.datetime`[4], the scan start. Only the solar zenith angle uses it, and the terminator blend is ~30° wide, so scan start is accurate enough.

`shachen.io.satellite.load_scene()` builds exactly this from ABI netCDF or AHI HSD files. The mapping from role to channel lives in *`shachen.constants.ABI_BANDS`* and *`shachen.constants.AHI_BANDS`*; if you read L1b some other way, produce the same variable names and you can skip the `satellite` extra entirely.

2.1.3 Running the algorithm

`shachen.pipeline.run_debra()` chains background → cloud mask → dust tests → confidence and returns every intermediate field alongside the answer:

```
import shachen

result = shachen.run_debra(scene, skin_temperature=merra_ts, emissivity=camel)

result["cf_comb"]  # combined dust confidence, 0–1 (Eq. 22)
result["dt1"]  # split-window test against the dynamic background
result["cm_norm_day"]  # normalised daytime cloud mask
result["zenith_deg"]  # per-pixel solar zenith
```

`skin_temperature` is MERRA-2 `TS` (K) on its native lat/lon grid; the pipeline regrids it via *`shachen.geo.regrid_latlon()`*. Pixels with NaN inputs (off-disk, bad pixels) carry NaN confidence throughout.

Choosing a background

The clear-sky background is estimated per pixel, by one of two schemes. Exactly one must be given: passing both, or neither, raises `ValueError`.

Scheme A — semi-analytic (paper §3.2, Option A). Per-band surface emissivity modifies the Planck radiance of the MERRA-2 skin temperature, and that is inverted back to a brightness temperature. Pass `emissivity=`, a CAMEL Dataset on its native grid:

[3] https://pyresample.readthedocs.io/en/stable/api/pyresample.geometry.html#pyresample.geometry.AreaDefinition
[4] https://docs.python.org/3/library/datetime.html#datetime.datetime

```
result = shachen.run_debra(scene, skin_temperature=merra_ts, emissivity=camel)
```

Scheme B — cloud-cleared composite (Option B). Stack the same time-of-day scenes from the preceding ~14 days and keep, per pixel, the bands from the day with the warmest `bt_tir_104`. Build it yourself and pass `background=`, already on the scene grid:

```
from shachen.composite import composite_background

bg = composite_background([day_1, day_2, ...])  # same grid, same hour
result = shachen.run_debra(scene, skin_temperature=merra_ts, background=bg)
```

Scheme B is built from real observations, so it carries the split-window water-vapour depression that the atmosphere-free scheme A lacks: the ~1 K high bias that zeroes DT1/DT2 on transparent winter plumes. Its per-pixel candidate count is passed through to the output as `n_valid`.

Any Dataset carrying `rsw_bg`, `btd_bg` and `bt_bg_tir_86/104/123` on the scene grid works as `background=`; `composite_background` and *shachen.background.background_signals()* both produce that contract.

Tuning

shachen.constants.DEFAULTS is the paper-tuned constant set, and it is the single source of every bound, offset and weight — each one appears in *Equations* in the formula it belongs to. Pass your own *shachen.constants.DebraConstants* to retune:

```
from shachen.constants import ABI_TUNED

result = shachen.run_debra(..., constants=ABI_TUNED)
```

shachen.constants.ABI_TUNED is the only shipped retune. It raises the Eq. 19 lower bound from 0.25 to 0.40 to suppress a clear-sky DT3 floor specific to the ABI + MERRA-2 + CAMEL stack. The reasoning and the numbers are in *Deviations*.

2.1.4 Enhanced imagery

The confidence field is rendered in two steps (Eqs. 23–29): a day/night blended greyscale baseline, then a per-gun colour modulation by `cf_comb`.

```
from shachen.imagery import debra_imagery, to_uint8
from shachen.render import render_debra_png

imagery = debra_imagery(scene, result)  # vis_bg, ir_bg, b_bg, bi, rgb
render_debra_png(
    to_uint8(imagery["rgb"]),
    scene.attrs["area"],
    scene.attrs["start_time"],
    Path("dust.png"),
)
```

Dust is yellow by default. The paper's other presets live in *shachen.constants.COLOR_DIMMING* and are selected with the `dimming=` argument:

```
imagery = debra_imagery(scene, result, dimming=COLOR_DIMMING["pink"])
```

`render_debra_png` is the only function that needs `cartopy` and `matplotlib`; both are imported inside it, so the rest of the package stays importable without the `render` extra.

2.1.5 Reproducing a reference case

End to end reproduction needs `[all]` plus an Earthdata[5] login in `~/.netrc`: MERRA-2 and CAMEL are authenticated downloads, while GOES L1b on AWS S3 is anonymous:

```
python scripts/fetch_case.py 2017-03-23-swus   # the paper's Figure 6 case
python scripts/run_case.py   2017-03-23-swus   # → netCDF + PNG
```

2.2 Equations

Every numbered equation of Miller et al. (2017), in the form this package implements: the 26 February 2020 erratum for Eqs. 7, 21, 22, 24 and 25, and the three prose-based corrections argued out in *Deviations*, each flagged below.

Brightness temperatures are written T_λ for the band centred near λ μm; θ is the solar zenith angle.

2.2.1 The normalization primitive

Almost every test is a clipped linear ramp between a MIN and a MAX bound (Eq. 3), which is where the tuning constants enter:

$$N(x; x_{\min}, x_{\max}) = \mathrm{clip}\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}},\ 0,\ 1\right)$$

`MIN` may exceed `MAX`, which reverses the ramp; the cos-zenith blends below rely on that. → *`shachen.norm.normalize()`*

2.2.2 Clear-sky background (§3.2)

The dynamic background separates DEBRA from a fixed-threshold test. It is estimated by one of two schemes.

Scheme A, semi-analytic. Surface emissivity ε_λ modifies the Planck radiance of the reanalysis skin temperature, inverted back to a brightness temperature:

$$T_\lambda^{\mathrm{bg}} = B^{-1}(\lambda,\ \varepsilon_\lambda\, B(\lambda, T_{\mathrm{skin}}))\,, \qquad B(\lambda, T) = \frac{c_1/\lambda^5}{\exp(c_2/\lambda T) - 1}$$

with $c_1 = 2hc^2$ and $c_2 = hc/k$. Missing emissivity (ocean) is treated as $\varepsilon = 1$, which makes the transform the identity. → *`shachen.background.background_signals()`*

Scheme B, cloud-cleared composite. Over a stack of n same-time-of-day scenes, take the warmest window pixel and read all three bands from that same day d^*. Clouds are cold, so the warmest day is taken as the clear-sky estimate:

$$d^* = \arg\max_{d \in \mathcal{C}}\ T_{10.4}^{(d)}, \qquad T_\lambda^{\mathrm{bg}} = T_\lambda^{(d^*)}$$

[5] https://urs.earthdata.nasa.gov/

where $\mathcal{C}$ is the set of days finite in all three bands at that pixel. → *shachen.composite.composite_background()*

Either way, the two background signals fed to the dust tests are

$$\mathrm{RSW}_{\mathrm{bg}} = T_{12.3}^{\mathrm{bg}} - T_{10.4}^{\mathrm{bg}}, \qquad \mathrm{BTD}_{\mathrm{bg}} = T_{8.6}^{\mathrm{bg}} - T_{10.4}^{\mathrm{bg}}$$

2.2.3 Cloud mask (Eqs. 1–12)

Four continuous cloud tests, damped by two dust-restoral terms so that dust is not masked away as cloud. → *shachen.cloudmask.cloud_mask()*

$$\mathrm{CM1} = 1 - N(T_{10.4};\ T_{\mathrm{skin}} - 50,\ T_{\mathrm{skin}}) \qquad \text{(Eqs. 1–2, cold relative to skin)}$$
$$\mathrm{CM2} = 1 - N(T_{10.4} - T_{6.2};\ 0,\ 25) \qquad \text{(Eq. 4, deep convection)}^{\dagger}$$
$$\mathrm{CM3} = N(T_{10.4} - T_{12.3};\ 2.0,\ 4.5) \qquad \text{(Eq. 5, thin cirrus, day + night)}$$
$$\mathrm{CM4} = N(T_{3.9} - T_{10.4};\ 5.0,\ 8.0) \qquad \text{(Eq. 6, thin cirrus, night only)}$$

The restoral terms: a pixel that looks like dust in the reverse split window subtracts from the cloud confidence.

$$R_1 = N(T_{12.3} - T_{10.4};\ 0,\ 3.5)\,(1 - \mathrm{CM1}) \qquad \text{(Eq. 7, erratum)}$$
$$R_2^{\mathrm{day}} = N(T_{8.6} - T_{10.4};\ -1,\ 3)\,(1 - \mathrm{CM2})(1 - \mathrm{CM3}) \qquad \text{(Eq. 8)}$$
$$R_2^{\mathrm{ngt}} = N(T_{8.6} - T_{10.4};\ -1,\ 3)\,(1 - \mathrm{CM2})(1 - \mathrm{CM4}) \qquad \text{(Eq. 9)}$$

Combined, then put on a common scale:

$$\mathrm{CM}^{\mathrm{ngt}} = (\mathrm{CM1} + \mathrm{CM2} + \mathrm{CM4})\left(1 - \max(R_1, R_2^{\mathrm{ngt}})\right) \qquad \text{(Eq. 10)}$$
$$\mathrm{CM}^{\mathrm{day}} = (\mathrm{CM1} + \mathrm{CM2} + \mathrm{CM3})\left(1 - \max(R_1, R_2^{\mathrm{day}})\right) \qquad \text{(Eq. 11)}^{\ddagger}$$
$$\mathrm{CM}_{\mathrm{norm}} = N(\mathrm{CM};\ 0.45,\ 0.80) \qquad \text{(Eq. 12)}$$

† Eq. 4 is implemented magnitude-reversed; as printed it saturates the mask over clear sky. ‡ Eq. 11 uses CM3 in place of the misprinted CM4: the 3.9 μm test is night-only. Both are argued in *Deviations*.

2.2.4 Dust tests (Eqs. 13–15)

DT1 and DT2 are the same ramp as Eq. 3 with the per-pixel background as the MIN bound; that substitution is what makes the test dynamic. Writing the observed signals $\mathrm{RSW} = T_{12.3} - T_{10.4}$ and $\mathrm{BTD} = T_{8.6} - T_{10.4}$:

$$\mathrm{DT1} = \mathrm{clip}\left(\frac{\mathrm{RSW} - \mathrm{RSW}_{\mathrm{bg}}}{3.5 - \mathrm{RSW}_{\mathrm{bg}}},\ 0,\ 1\right) \qquad \text{(Eq. 13)}$$
$$\mathrm{DT2} = \mathrm{clip}\left(\frac{\mathrm{BTD} - \mathrm{BTD}_{\mathrm{bg}}}{3.0 - \mathrm{BTD}_{\mathrm{bg}}},\ 0,\ 1\right) \qquad \text{(Eq. 14)}$$

DT3 is the thermal-contrast test, implemented magnitude-reversed per the paper's prose ("observations that are relatively cold compared to MERRA produce high value for DT3"):

$$\mathrm{DT3} = \mathrm{clip}\left(\frac{(T_{\mathrm{MERRA}} - S) - T_{10.4}}{50},\ 0,\ 1\right), \qquad S = \begin{cases} -10\ \mathrm{K} & \text{land} \\ +5\ \mathrm{K} & \text{ocean} \end{cases}$$

→ *shachen.dust_tests.dust_tests()*

2.2.5 Confidence factor (Eqs. 16–22)

Three illumination regimes, each suppressed by the matching cloud mask. Night drops to $\max(\mathrm{DT1}, \mathrm{DT2})$ because the two split-window tests stop being independent without solar heating. → *shachen.confidence.confidence()*

$$\mathrm{CF}^{*}_{\mathrm{day}} = (\mathrm{DT1} + \mathrm{DT2} + \mathrm{DT3})\left(1 - \mathrm{CM}^{\mathrm{day}}_{\mathrm{norm}}\right) \qquad \text{(Eq. 16)}$$

$$\mathrm{CF}^{*}_{\mathrm{trm}} = (\mathrm{DT1} + \mathrm{DT2} + \tfrac{1}{2}\mathrm{DT3})\left(1 - \mathrm{CM}^{\mathrm{day}}_{\mathrm{norm}}\right) \qquad \text{(Eq. 17)}$$

$$\mathrm{CF}^{*}_{\mathrm{ngt}} = \left(\max(\mathrm{DT1}, \mathrm{DT2}) + \tfrac{1}{2}\mathrm{DT3}\right)\left(1 - \mathrm{CM}^{\mathrm{ngt}}_{\mathrm{norm}}\right) \qquad \text{(Eq. 18)}$$

$$\mathrm{CF} = N(\mathrm{CF}^{*};\ 0.25,\ 2.50) \qquad \text{(Eq. 19)}$$

The terminator is crossed smoothly, with weights evaluated in cosine-zenith space:

$$B^{\mathrm{trm}}_{\mathrm{ngt}} = N(\cos\theta;\ \cos 105^\circ,\ \cos 90^\circ)^{1.5} \qquad \text{(Eq. 20)}$$

$$B^{\mathrm{day}}_{\mathrm{trm}} = N(\cos\theta;\ \cos 90^\circ,\ \cos 75^\circ)^{1.5} \qquad \text{(Eq. 21, erratum)}$$

$$\mathrm{CF}_{\mathrm{comb}} = B^{\mathrm{day}}_{\mathrm{trm}}\,\mathrm{CF}_{\mathrm{day}} + \left(1 - B^{\mathrm{day}}_{\mathrm{trm}}\right)\left[B^{\mathrm{trm}}_{\mathrm{ngt}}\,\mathrm{CF}_{\mathrm{trm}} + \left(1 - B^{\mathrm{trm}}_{\mathrm{ngt}}\right)\mathrm{CF}_{\mathrm{ngt}}\right] \qquad \text{(Eq. 22, erratum)}$$

$\mathrm{CF}_{\mathrm{comb}} \in [0, 1]$ is the algorithm's output field.

2.2.6 Enhanced imagery (Eqs. 23–29)

A greyscale baseline image, blended day-to-night, then modulated by the confidence factor in each colour gun. Domain min/max are taken over the whole scene, skipping NaN. → *shachen.imagery.debra_imagery()*

$$\mathrm{VIS}_{\mathrm{bg}} = \frac{\rho - \rho_{\min}}{\rho_{\max} - \rho_{\min}} \qquad \text{(Eq. 23)}$$

$$\mathrm{IR}_{\mathrm{bg}} = 1 - \frac{T_{10.4} - T_{\min}}{T_{\max} - T_{\min}} \qquad \text{(Eq. 24, erratum)}$$

$$B_{\mathrm{bg}} = 1 - N(\theta;\ 79^\circ,\ 89^\circ)^{1.5} \qquad \text{(Eq. 25, erratum)}$$

$$\mathrm{BI} = B_{\mathrm{bg}}\,\mathrm{VIS}_{\mathrm{bg}} + \left(1 - B_{\mathrm{bg}}\right)\mathrm{IR}_{\mathrm{bg}} \qquad \text{(Eq. 26)}$$

Eq. 25 is evaluated in zenith degree space, unlike the cos-space Eqs. 20–21. Each gun then gets the same form, differing only in the coefficient D_G applied to the confidence factor:

$$G = N(\mathrm{BI}\,(1 - \min(\mathrm{CF}_{\mathrm{comb}}, 0.5)) + D_G\,\mathrm{CF}_{\mathrm{comb}};\ 0,\ 1.2)\,, \qquad G \in \{R, G, B\}$$

The printed Eqs. 27–29 are $D_R = D_G = 1$ and $D_B = 0.10$, which paints dust yellow. The paper's §4.2 alternatives are generalised in *shachen.constants.COLOR_DIMMING*:

Preset	(D_R, D_G, D_B)
`yellow` (Eqs. 27–29)	$(1.0,\ 1.0,\ 0.10)$
`pink`	$(1.0,\ 0.25,\ 0.25)$
`green`	$(0.10,\ 1.0,\ 0.10)$
`blue`	$(0.25,\ 0.25,\ 1.0)$

The $\min(\mathrm{CF}, 0.5)$ cap keeps dust translucent: even at full confidence, half the underlying scene still shows through.

2.3 API reference

Every DEBRA module maps onto a numbered block of equations from Miller et al. (2017). Read down the table to follow the algorithm in the order it runs; the *Equations* page writes those equations out in full. The one entry marked "—" under Equations is the classic Dust RGB baseline, which is not part of the paper.

Equations	Module	What it does
—	*shachen.io*	L1b → calibrated fields; MERRA-2 and CAMEL ancillary
—	*shachen.geo*	lat/lon ancillary → satellite grid, land mask
—	*shachen.solar*	per-pixel solar zenith angle
Eq. 3	*shachen.norm*	the normalisation primitive every test uses
§3.2 A	*shachen.background*	semi-analytic clear-sky background
§3.2 B	*shachen.composite*	cloud-cleared composite background
Eqs. 1–12	*shachen.cloudmask*	cloud confidence with dust restoral
Eqs. 13–15	*shachen.dust_tests*	DT1–DT3 against the dynamic background
Eqs. 16–22	*shachen.confidence*	day/terminator/night blend → `cf_comb`
Eqs. 1–22	*shachen.pipeline*	the entry point per algorithm
Eqs. 23–29	*shachen.imagery*	baseline image and CF-modulated RGB
§4.2	*shachen.render*	georeferenced PNG with map overlays
—	*shachen.dustrgb*	classic Dust RGB comparison baseline
all	*shachen.constants*	every bound, offset and weight

2.3.1 Top-level namespace

shachen: infrared-channel dust algorithms for geostationary imagers.

DEBRA-Dust, the Dynamic Enhancement with Background Reduction Algorithm of Miller et al. (2017), doi:10.1002/2017JD027365 (with the 26 Feb 2020 erratum), is the primary one; the classic EUMETSAT Dust RGB ships alongside it as the baseline to compare against.

class shachen.**Band**

Bases: `StrEnum`[6]

Spectral roles the algorithms read (nominal wavelengths in um).

Seven of these are DEBRA's inputs (`DEBRA_BANDS`); `TIR_112` exists only for the classic Dust RGB green gun.

VIS_064 = 'vis_064'

cloud mask + day baseline image

NIR_160 = 'nir_160'

daytime cloud test (reserved)

SWIR_39 = 'swir_39'

night thin-cirrus test CM4

WV_62 = 'wv_62'

deep-convection test CM2

TIR_86 = 'tir_86'

dust test DT2 (8.4-8.6 um)

TIR_104 = 'tir_104'

clean window reference (10.3-10.4 um)

TIR_112 = 'tir_112'

classic Dust RGB green gun only (11.2 um; not a DEBRA input)

TIR_123 = 'tir_123'

dirty window, RSW / DT1 (12.3 um)

__new__(*value*)

shachen.**normalize**(*x*, *bounds:* Bounds)

N(x) = clip((x - MIN) / (MAX - MIN), 0, 1) (Miller et al. 2017, Eq. 3).

Works on scalars, numpy arrays, and xarray DataArrays. `bounds.min` may exceed `bounds.max` (used for the cos-zenith blends specified in zenith-angle space): the sense of the ramp reverses.

shachen.**run_debra**(*scene:* Dataset[7], *skin_temperature:* DataArray[8], *emissivity:* Dataset[9] | None[10] = None, *constants:* DebraConstants = DebraConstants(cloud_mask=CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8)), dust_tests=DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0), confidence=ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.25, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0)), imagery=ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2)), *, *(Keyword-only parameters separator (PEP 3102))*, *background:* Dataset[11] | None[12] = None) → Dataset[13]

Run DEBRA on one scene; returns CF_comb plus all intermediate fields.

`scene` is a *`shachen.io.satellite.load_scene()`* Dataset (`bt_*` in K on the 2-km grid, with `area` and `start_time` attrs); `skin_temperature` is MERRA-2 TS (K) on its native lat/lon grid, regridded here via *`shachen.geo.regrid_latlon()`*. The visible/NIR reflectance variables are not used here; they feed the enhanced imagery.

Exactly one background source must be given (ValueError otherwise):

- `emissivity`: the CAMEL band Dataset (`emis_*`) on its native lat/lon grid; regridded here, then fed through *`shachen.background.background_signals()`* (semianalytic mode);
- `background`: a precomputed Dataset **already on the scene grid** (e.g. *`shachen.composite.composite_background()`*) carrying `rsw_bg`, `btd_bg` and `bt_bg_tir_86/104/123`; missing variables or 2-D shapes differing from the scene raise ValueError. Its `n_valid` is passed through to the output when present.

Returns a Dataset on the scene grid carrying `cf_comb`, `cf_day`, `cf_trm`, `cf_ngt`, `cm_norm_day`, `cm_norm_ngt`, `dt1-dt3`, `rsw_bg`, `btd_bg`, and `zenith_deg`, with the scene's `area` and `start_time` attrs preserved. Pixels with NaN inputs (off-disk, bad pixels) carry NaN confidence.

[6] https://docs.python.org/3/library/enum.html#enum.StrEnum

shachen.**run_dust_rgb**(*scene:* *Dataset*[14], *constants:* DustRGBConstants | *None*[15] = *None*) → Dataset[16]

Run the classic Dust RGB baseline on one scene.

The counterpart of `run_debra()` for the recipe in *shachen.dustrgb*: same scene in, but no ancillary data, no cloud mask and no confidence field — three fixed stretches of `bt_tir_86/104/112/123` (11.2 um is the extra band DEBRA itself never reads). A scene loaded with `roles=DEBRA_BANDS` therefore raises ValueError here.

The stretches are per sensor. Unlike DEBRA, the Dust RGB has no one canonical set of numbers: it was tuned for SEVIRI and then re-tuned for each later imager, because the corresponding channels do not sit at the same wavelengths. With `constants=None` (the default) the set is chosen from `scene.attrs["reader"]` through *shachen.constants.DUST_RGB_BY_READER* — ABI gets the Quick Guide's adjusted values, AHI the original SEVIRI ones — so the baseline matches that sensor's operational product. An unknown or absent reader falls back to *shachen.constants.DUST_RGB* (SEVIRI); pass `constants` explicitly to pin one set across sensors, e.g. to compare the two.

Returns a Dataset carrying `dust_rgb` — dims `(y, x, gun)`, floats in [0, 1], ready for *shachen.imagery.to_uint8()* — with the scene's `area` and `start_time` attrs preserved, so it merges straight into a `run_debra()` result for side-by-side rendering.

Pipeline

End-to-end DEBRA run, Eqs. 1-22. End-to-end entry points: one call per algorithm, scene in, fields out.

run_debra() is DEBRA, Eqs. 1-22: regrid ancillary onto the scene grid, derive solar zenith and land mask, then chain background -> cloud mask -> dust tests -> confidence. Enhanced imagery (Eqs. 23-29) is in *shachen.imagery*.

run_dust_rgb() is the classic Dust RGB baseline, which needs no ancillary data at all but does need to know which sensor it is looking at. Callers reach both through this module rather than the per-equation modules, so adding an algorithm does not change how one is run.

[7] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[8] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[9] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[10] https://docs.python.org/3/library/constants.html#None
[11] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[12] https://docs.python.org/3/library/constants.html#None
[13] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[14] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[15] https://docs.python.org/3/library/constants.html#None
[16] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

shachen.pipeline.**run_debra**(*scene: Dataset*[17], *skin_temperature: DataArray*[18], *emissivity: Dataset*[19] | *None*[20] = *None*, *constants:* DebraConstants = *DebraConstants(cloud_mask=CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8)), dust_tests=DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0), confidence=ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.25, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0)), imagery=ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2))*, *, *background: Dataset*[21] | *None*[22] = *None*) → Dataset[23]

Run DEBRA on one scene; returns CF_comb plus all intermediate fields.

`scene` is a *`shachen.io.satellite.load_scene()`* Dataset (`bt_*` in K on the 2-km grid, with `area` and `start_time` attrs); `skin_temperature` is MERRA-2 TS (K) on its native lat/lon grid, regridded here via *`shachen.geo.regrid_latlon()`*. The visible/NIR reflectance variables are not used here; they feed the enhanced imagery.

Exactly one background source must be given (ValueError otherwise):

- `emissivity`: the CAMEL band Dataset (`emis_*`) on its native lat/lon grid; regridded here, then fed through *`shachen.background.background_signals()`* (semianalytic mode);
- `background`: a precomputed Dataset **already on the scene grid** (e.g. *`shachen.composite.composite_background()`*) carrying `rsw_bg`, `btd_bg` and `bt_bg_tir_86/104/123`; missing variables or 2-D shapes differing from the scene raise ValueError. Its `n_valid` is passed through to the output when present.

Returns a Dataset on the scene grid carrying `cf_comb`, `cf_day`, `cf_trm`, `cf_ngt`, `cm_norm_day`, `cm_norm_ngt`, `dt1-dt3`, `rsw_bg`, `btd_bg`, and `zenith_deg`, with the scene's `area` and `start_time` attrs preserved. Pixels with NaN inputs (off-disk, bad pixels) carry NaN confidence.

shachen.pipeline.**run_dust_rgb**(*scene: Dataset*[24], *constants:* DustRGBConstants | *None*[25] = *None*) → Dataset[26]

Run the classic Dust RGB baseline on one scene.

The counterpart of *`run_debra()`* for the recipe in *`shachen.dustrgb`*: same scene in, but no ancillary data, no cloud mask and no confidence field — three fixed stretches of `bt_tir_86/104/112/123` (11.2 um is the extra band DEBRA itself never reads). A scene loaded with `roles=DEBRA_BANDS` therefore raises ValueError here.

The stretches are per sensor. Unlike DEBRA, the Dust RGB has no one canonical set of numbers: it

[17] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[18] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[19] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[20] https://docs.python.org/3/library/constants.html#None
[21] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[22] https://docs.python.org/3/library/constants.html#None
[23] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

was tuned for SEVIRI and then re-tuned for each later imager, because the corresponding channels do not sit at the same wavelengths. With `constants=None` (the default) the set is chosen from `scene.attrs["reader"]` through *shachen.constants.DUST_RGB_BY_READER* — ABI gets the Quick Guide's adjusted values, AHI the original SEVIRI ones — so the baseline matches that sensor's operational product. An unknown or absent reader falls back to *shachen.constants.DUST_RGB* (SEVIRI); pass `constants` explicitly to pin one set across sensors, e.g. to compare the two.

Returns a Dataset carrying `dust_rgb` — dims `(y, x, gun)`, floats in [0, 1], ready for *shachen.imagery.to_uint8()* — with the scene's `area` and `start_time` attrs preserved, so it merges straight into a *run_debra()* result for side-by-side rendering.

Background (semi-analytic)

Emissivity x Planck clear-sky background, paper section 3.2 Option A. Clear-sky background reference signals (Miller et al. 2017, section 3.2).

The semianalytic background: per-band surface emissivity (CAMEL climatology, interpolated to the sensor band centers) modifies the Planck blackbody radiance of the MERRA-2 skin temperature, which is inverted back to an equivalent brightness temperature: `BT_bg = B^-1(eps * B(T_skin))`. Band differences of these give the dynamic backgrounds used by the dust tests (Eqs. 13-14): `RSW_bg = BT_bg(12.3) - BT_bg(10.4)` and `BTD_bg = BT_bg(8.6) - BT_bg(10.4)`.

The Planck function is monochromatic at the band centers (`BAND_CENTER_UM`), with physical constants from scipy.constants; only background *differences* enter DEBRA, so the monochromatic approximation is well inside the paper's quoted <5-10% MERRA-driven uncertainty on the final confidence factor.

Missing emissivity (NaN, e.g. ocean in CAMEL) is treated as eps = 1.0: the paper notes water-surface infrared emissivity is close to unity and needs no background correction, so the tests fall back to their static bounds there.

`shachen.background.`**`planck_radiance`**(*temperature_k*, *wavelength_um*)

Monochromatic Planck spectral radiance B(lambda, T) in W m-2 sr-1 m-1.

Array-generic (scalars, numpy, xarray); `wavelength_um` in micrometres.

`shachen.background.`**`planck_temperature`**(*radiance*, *wavelength_um*)

Analytic inverse of *planck_radiance()*: brightness temperature in K.

`shachen.background.`**`background_bt`**(*skin_temperature*, *emissivity*, *band:* Band)

Background brightness temperature `B^-1(eps * B(T_skin))` for one band.

Array-generic; NaN emissivity is treated as 1.0 (identity: BT_bg = T_skin).

`shachen.background.`**`background_signals`**(*skin_temperature: DataArray*[27], *emissivity: Dataset*[28]) → Dataset[29]

Per-pixel background signals for the dust tests (Eqs. 13-14).

`emissivity` must carry `emis_tir_86`, `emis_tir_104`, `emis_tir_123` (the `load_band_emissivity` naming), on the same grid as `skin_temperature` (K). Raises ValueError if the 2-D shapes differ.

Returns a Dataset with `rsw_bg` (= bt_bg_tir_123 - bt_bg_tir_104), `btd_bg` (= bt_bg_tir_86 - bt_bg_tir_104), and the per-band `bt_bg_tir_86`, `bt_bg_tir_104`, `bt_bg_tir_123` for debugging.

[24] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[25] https://docs.python.org/3/library/constants.html#None
[26] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

Background (composite)

Cloud-cleared multiday composite background, paper section 3.2 Option B. Cloud-cleared composite clear-sky background (Miller et al. 2017, §3.2).

The emissivity-free alternative to *shachen.background*: stack the same time-of-day scenes from the preceding ~14 days and, per pixel, keep the three TIR brightness temperatures from the day with the warmest BT10.4 (clouds are cold; the warmest day is taken as the clear-sky estimate). Because the composite is built from real observations it carries the split-window water-vapor depression the semianalytic (emissivity x Planck, no atmosphere) background lacks: the ~1 K high bias that zeroes DT1/DT2 on transparent winter plumes (see docs/deviations.md).

Spectral coherence: all three bands come from the same selected day, and a day is a candidate at a pixel only where all three bands are finite there.

shachen.composite.**COMPOSITE_BANDS: tuple**[30]**[*Band*, ...] = (Band.TIR_86, Band.TIR_104, Band.TIR_123)**

The bands a composite scene must carry (the three DEBRA TIR windows).

shachen.composite.**composite_background**(*scenes: Sequence*[31]*[Dataset*[32]*]*) → Dataset[33]

Clear-sky background signals from a stack of same-time-of-day scenes.

Each element of `scenes` must carry 2-D `bt_tir_86`, `bt_tir_104`, `bt_tir_123` (K) on one common grid; ValueError on an empty sequence, a missing variable, or any shape mismatch. Per pixel, a scene is a *candidate* only where all three bands are finite; the candidate with the warmest `bt_tir_104` is selected (first wins on exact ties) and all three bands are taken from it. Pixels with zero candidates are NaN.

Returns the *shachen.background.background_signals()* contract — `bt_bg_tir_86`, `bt_bg_tir_104`, `bt_bg_tir_123` (units "K"), `rsw_bg` (= bt_bg_tir_123 - bt_bg_tir_104) and `btd_bg` (= bt_bg_tir_86 - bt_bg_tir_104) — plus `n_valid` (per-pixel candidate count, integer dtype) and `attrs["n_scenes"] = len(scenes)`.

Cloud mask

Cloud confidence with dust restoral, Eqs. 1-12. DEBRA cloud mask, Eqs. 1-12 (Miller et al. 2017).

Continuous cloud confidence in [0, 1] from IR-only tests, with dust-restoral terms so dust pixels are not masked as cloud. Two deliberate deviations from the *printed* equations, both following the paper's own prose (the 2020 erratum does not touch either); see docs/deviations.md:

- Eq. 4 (CM2) is implemented magnitude-reversed, `CM2 = 1 - N(BT10.4 - BT6.2; 0, 25)` (deep convection -> 1, clear -> 0). As printed, clear sky (split of 25-55 K) would give CM2 ~ 1 and saturate the mask everywhere, degenerating the algorithm; the prose ("in a similar fashion" to the reversed Eq. 1) and Figure 3c confirm the reversed intent.
- Eq. 11 (CM_day) uses CM3, not the misprinted CM4 (the 3.9-um test is night-only): `CM_day = (CM1 + CM2 + CM3) * (1 - max(R1, R2_day))`.

[27] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[28] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[29] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[30] https://docs.python.org/3/library/stdtypes.html#tuple
[31] https://docs.python.org/3/library/collections.abc.html#collections.abc.Sequence
[32] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[33] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

shachen.cloudmask.**cloud_mask**(*scene: Dataset*[34], *skin_temperature: DataArray*[35], *constants:* CloudMaskConstants = *CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8))*) → Dataset[36]

Cloud-mask components and combined masks on the scene grid.

`scene` needs `bt_swir_39`, `bt_wv_62`, `bt_tir_86`, `bt_tir_104`, `bt_tir_123` (K); `skin_temperature` (K) must be on the same grid (ValueError on 2-D shape mismatch). NaN inputs propagate.

Returns a Dataset with `cm1`..``cm4`` (Eqs. 1, 4-6; CM1 reduces to `1 - N(BT10.4 - T_skin; -offset, 0)` since the per-pixel bounds share a constant width), `r1`, `r2_day`, `r2_ngt` (Eqs. 7-9), `cm_day`, `cm_ngt` (Eqs. 10-11), and `cm_norm_day`, `cm_norm_ngt` (Eq. 12).

Dust tests

DT1-DT3 against the dynamic background, Eqs. 13-15. DEBRA dust detection tests DT1-DT3, Eqs. 13-15 (Miller et al. 2017).

DT1/DT2 measure the observed split-window signals against the per-pixel *dynamic* background (RSW_bg / BTD_bg used as the MIN bound of Eq. 3). DT3 is the thermal-contrast consensus builder; it is implemented magnitude-reversed relative to the printed Eq. 15, per the paper's prose ("observations that are relatively cold compared to MERRA produce high value for DT3"); see docs/deviations.md:

DT3 = clip(((T_MERRA - S) - BT10.4) / depth, 0, 1)

with the surface shift S = -10 K (land) / +5 K (ocean) applied as printed (`T_MERRA - S`) and depth = 50 K.

shachen.dust_tests.**dt1**(*rsw_obs*, *rsw_bg*, *constants:* DustTestConstants = *DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0)*)

Eq. 13: `(RSW_obs - RSW_bg) / (MAX_RSW - RSW_bg)` clipped to [0, 1].

`RSW = BT12.3 - BT10.4` (K). Array-generic. Where the dynamic range is degenerate (`RSW_bg >= MAX_RSW`) the test returns 0.0; NaN propagates.

shachen.dust_tests.**dt2**(*btd_obs*, *btd_bg*, *constants:* DustTestConstants = *DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0)*)

Eq. 14: like *dt1()* for `BTD = BT8.6 - BT10.4` with MAX = 3.0 K.

shachen.dust_tests.**dt3**(*bt_104*, *skin_temperature*, *is_land*, *constants:* DustTestConstants = *DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0)*)

Eq. 15 (magnitude-reversed, see module docstring).

`is_land` selects the land/ocean surface shift. Array-generic; NaN in `bt_104` or `skin_temperature` propagates.

[34] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[35] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[36] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

shachen.dust_tests.**dust_tests**(*scene: Dataset*[37], *background: Dataset*[38], *skin_temperature: DataArray*[39], *is_land: DataArray*[40], *constants:* DustTestConstants *= DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0)*) → Dataset[41]

Assemble `dt1`, `dt2`, `dt3` fields on the scene grid.

`scene` needs `bt_tir_86`, `bt_tir_104`, `bt_tir_123`; `background` needs `rsw_bg`, `btd_bg` (from *`shachen.background.background_signals()`*). All 2-D inputs must share one shape (ValueError otherwise). Returns a Dataset with `dt1`, `dt2`, `dt3` in [0, 1] (NaN where inputs are NaN).

Confidence factor

Day / terminator / night blend into CF_comb, Eqs. 16-22. DEBRA confidence factor, Eqs. 16-22 (Eqs. 21-22 per the 2020 erratum).

Combines the dust tests under cloud-mask suppression into day / terminator / night confidence factors (Eqs. 16-18), normalizes each with the (0.25, 2.50) bounds (Eq. 19), and blends them across the terminator with solar-zenith weights `B_ngt_trm` and `B_trm_day` (Eqs. 20-21, exponent 1.5) into the final `CF_comb` in [0, 1] (Eq. 22).

shachen.confidence.**confidence**(*tests: Dataset*[42], *cloud: Dataset*[43], *zenith_deg: DataArray*[44], *constants:* ConfidenceConstants *= ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.25, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0))*) → Dataset[45]

Confidence factors and blend weights on the scene grid.

`tests` needs `dt1`, `dt2`, `dt3` (from *`shachen.dust_tests.dust_tests()`*); `cloud` needs `cm_norm_day`, `cm_norm_ngt` (from *`shachen.cloudmask.cloud_mask()`*); `zenith_deg` is the solar zenith angle in degrees. All 2-D inputs must share one shape (ValueError otherwise); NaN propagates.

Returns a Dataset with `cf_day`, `cf_trm`, `cf_ngt` (each already normalized per Eq. 19; CF_trm and CF_ngt use `cm_norm_day` and `cm_norm_ngt` respectively, CF_ngt takes `max(DT1, DT2)`), the blend weights `b_ngt_trm`, `b_trm_day` (via *`shachen.norm.normalize_cos_zenith()`*), and `cf_comb` (Eq. 22).

Enhanced imagery

Baseline image and CF-modulated RGB, Eqs. 23-29. DEBRA enhanced imagery, Eqs. 23-29 (Eqs. 24-25 per the 2020 erratum).

[37] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[38] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[39] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[40] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[41] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[42] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[43] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[44] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[45] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

Builds the day/night blended baseline image (Eqs. 23-26) and modulates it by the combined confidence factor in each color gun (Eqs. 27-29). Pure array numerics; map rendering is in `shachen.render`.

`shachen.imagery.`**`baseline_image`**(*scene: Dataset*[46], *zenith_deg: DataArray*[47], *constants:* ImageryConstants = *ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2)*) → Dataset[48]

Day/night blended baseline image BI (Eqs. 23-26).

`scene` needs `refl_vis_064` (any linear reflectance unit; the Eq. 23 domain min/max normalization cancels units) and `bt_tir_104` (K); `zenith_deg` is the solar zenith angle in degrees. Missing variables and 2-D shape mismatches raise ValueError.

Returns a Dataset on dims `(y, x)` with

- `vis_bg` (Eq. 23): VIS normalized by the domain min/max reflectance;
- `ir_bg` (Eq. 24, erratum): inverted normalized BT10.4, 1.0 at the domain cold bound;
- `b_bg` (Eq. 25, erratum): `1 - N(zenith; 79, 89)**1.5` evaluated in zenith-*degree* space (unlike the cos-space Eqs. 20-21);
- `bi` (Eq. 26): `b_bg*vis_bg + (1 - b_bg)*ir_bg`, in [0, 1].

NaN semantics: NaN pixels propagate, except a component whose Eq. 26 blend weight is exactly zero is ignored (a NaN VIS pixel on the night side leaves `bi = ir_bg`). A degenerate component domain (all-NaN, or domain max == min, e.g. an all-dark night VIS band) yields 0.0 for that component everywhere instead of dividing by zero.

`shachen.imagery.`**`enhanced_rgb`**(*baseline: DataArray*[49], *cf_comb: DataArray*[50], *constants:* ImageryConstants = *ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2)*, *dimming: tuple*[51]*[float*[52]*, float*[53]*, float*[54]*]* | *None*[55] = *None*) → DataArray[56]

Color-modulated composite (Eqs. 27-29 + the Eq. 3 gun rescale).

Each gun is `BI*(1 - min(CF, cf_cap)) + D_gun*CF` with the per-gun dimming triple `dimming` (`None` selects `COLOR_DIMMING["yellow"]`, i.e. D = 1.0 on red/green and `blue_dimming` on blue), then rescaled with `N(gun; 0, gun_max)` onto [0, 1].

Returns a float DataArray with dims `("y", "x", "gun")` and coordinate `gun = ["r", "g", "b"]` (the dim is not named `rgb` so the array can live in a Dataset as variable `rgb` without a name collision). NaN propagates; a shape mismatch between `baseline` and `cf_comb` raises ValueError.

[46] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[47] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[48] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[49] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[50] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[51] https://docs.python.org/3/library/stdtypes.html#tuple
[52] https://docs.python.org/3/library/functions.html#float
[53] https://docs.python.org/3/library/functions.html#float
[54] https://docs.python.org/3/library/functions.html#float
[55] https://docs.python.org/3/library/constants.html#None
[56] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray

shachen.imagery.**to_uint8**(*rgb: DataArray*[57]) → ndarray[58]

Convert the [0, 1] float RGB composite to a `(y, x, 3)` uint8 array.

`rgb` is an *enhanced_rgb()*-shaped DataArray (gun axis last). Values are scaled by 255 and rounded with `numpy.rint()`; NaN pixels (off-disk) become 0 (black).

shachen.imagery.**debra_imagery**(*scene: Dataset*[59], *debra: Dataset*[60], *constants:* DebraConstants = *DebraConstants(cloud_mask=CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8)), dust_tests=DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0), confidence=ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.25, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0)), imagery=ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2))*, *dimming: tuple*[61]*[float*[62]*, float*[63]*, float*[64]*]* | *None*[65] = *None*) → Dataset[66]

Full imagery chain (Eqs. 23-29) from a scene and a run_debra output.

`scene` needs `refl_vis_064` and `bt_tir_104`; `debra` needs `cf_comb` and `zenith_deg` (from *shachen.pipeline.run_debra()*). Missing variables raise ValueError.

Returns a Dataset with the *baseline_image()* fields (`vis_bg`, `ir_bg`, `b_bg`, `bi`) plus `rgb` from *enhanced_rgb()*, with `scene`'s attrs preserved.

Render

Georeferenced PNG with coastlines and graticule, paper section 4.2. Render the DEBRA RGB composite to a georeferenced PNG with map overlays.

Paper section 4.2: "Georeferenced metainformation such as coastlines, political boundaries, latitude/longitude grid, and date/time information … are overlaid to the R/G/B imagery." Uses cartopy on the scene's projection, same styling as the quicklook renderer.

shachen.render.**render_debra_png**(*rgb: ndarray*[67], *area: AreaDefinition*[68], *when: datetime*[69], *out_path: Path*[70], *title: str*[71] | *None*[72] = *None*, *dpi: int*[73] = *150*) → Path[74]

[57] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[58] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[59] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[60] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[61] https://docs.python.org/3/library/stdtypes.html#tuple
[62] https://docs.python.org/3/library/functions.html#float
[63] https://docs.python.org/3/library/functions.html#float
[64] https://docs.python.org/3/library/functions.html#float
[65] https://docs.python.org/3/library/constants.html#None
[66] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

Write `rgb` as a PNG with coastlines, borders, and a lat/lon grid.

`rgb` must be a `(y, x, 3)` uint8 array (from *shachen.imagery.to_uint8()*) on `area`'s grid — anything else raises ValueError. `when` is the scan start time (UTC), used in the default title `"DEBRA-Dust — %Y-%m-%d %H:%M UTC"`. Renders on the Agg backend and returns `out_path`.

Classic Dust RGB

The standard operational infrared dust composite: three fixed band-difference stretches that make lofted mineral dust read pink to magenta over dark blue-green surfaces, day and night. Forecasters have used it on SEVIRI, ABI and AHI for years; it needs no ancillary data and no cloud mask, which is both why it is cheap to run and why it produces no per-pixel measure of dust amount. *shachen.pipeline.run_dust_rgb* is the entry point.

In this package it also serves as the comparison baseline for DEBRA-Dust: the image to put next to an enhancement to see what the enhancement changed.

Per-sensor stretches

The Dust RGB has no single canonical set of numbers. It was tuned for Meteosat SEVIRI and then re-tuned for each later imager, because the corresponding channels do not sit at the same wavelengths — most visibly in the green gun, where SEVIRI's `IR10.8 − IR8.7` becomes `11.2 − 8.4` on ABI. Berndt et al. (2018) is the method behind those adjustments, and the same reasoning produced ABI-specific versions of the Ash, Convection and Night Microphysics RGBs alongside Dust.

`run_dust_rgb` therefore reads `scene.attrs["reader"]` and picks the set that sensor's operational product uses, so a baseline rendered here matches the image forecasters see:

Reader	Constant	Red 12.3 − 10.3	Green 11.2 − 8.4 (γ 2.5)	Blue 10.3
`abi_l1b`	`DUST_RGB_ABI`	−6.7 to +2.6 K	−0.5 to +20.0 K	261.2 to 288.7 K
`ahi_hsd`	`DUST_RGB`	−4 to +2 K	0 to +15 K	261 to 289 K

Himawari gets the original SEVIRI values because no re-tuned AHI recipe has been published; that is also what satpy renders, since it ships a `dust_abi` enhancement but no `dust_ahi`. An unknown or absent reader falls back to the same SEVIRI set. Pass `constants=` explicitly to pin one set across sensors, which applies when comparing two sensors on identical numbers.

The bands themselves never change: `DEBRA_BANDS` plus 11.2 μm, the one channel DEBRA never reads.

[67] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[68] https://pyresample.readthedocs.io/en/stable/api/pyresample.geometry.html#pyresample.geometry.AreaDefinition
[69] https://docs.python.org/3/library/datetime.html#datetime.datetime
[70] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[71] https://docs.python.org/3/library/stdtypes.html#str
[72] https://docs.python.org/3/library/constants.html#None
[73] https://docs.python.org/3/library/functions.html#int
[74] https://docs.python.org/3/library/pathlib.html#pathlib.Path

Citing this baseline

This is a published scheme in its own right, cited independently of anything else in this package. If a figure or a number in your work comes from `run_dust_rgb`, cite Lensky and Rosenfeld (2008) plus the recipe for the sensor you ran it on — the Quick Guide for ABI, the EUMeTrain compilation for AHI. The repository's `CITATION.cff` records each reference with that scope, and the README maps every algorithm in the package to what it needs cited.

References

- Lensky, I. M., and D. Rosenfeld (2008): Clouds-Aerosols-Precipitation Satellite Analysis Tool (CAPSAT). *Atmos. Chem. Phys.*, **8**, 6739–6753. doi:10.5194/acp-8-6739-2008[75] — the SEVIRI RGB suite this scheme comes from.
- EUMeTrain: Compilation of RGB Recipes[76] — the SEVIRI Dust RGB as formalised for operations; the source of `DUST_RGB`.
- NOAA/NASA GOES-R Quick Guide: Dust RGB[77] (contributor K. Fuell, NASA SPoRT; CIRA/RAMMB) — the ABI band mix, and the source of `DUST_RGB_ABI`.
- Berndt, E., N. Elmer, L. Schultz, and A. Molthan (2018): A Methodology to Determine Recipe Adjustments for Multispectral Composites Derived from Next-Generation Advanced Satellite Imagers. *J. Atmos. Oceanic Technol.*, **35**, 643–664. doi:10.1175/JTECH-D-17-0047.1[78] — why a recipe has to be re-tuned per imager.

Classic Dust RGB — the standard operational infrared dust composite.

Three fixed stretches of the infrared channels, no ancillary data and no cloud mask, which is why it is cheap and why it produces no per-pixel measure of dust amount. Not part of the DEBRA algorithm (Eqs. 1-29); here it is the baseline an enhancement gets compared against. RED = BT12.3 - BT10.4, GRN = BT11.2 - BT8.6 (gamma 2.5), BLU = BT10.4. The stretch values are per sensor: the scheme was tuned for SEVIRI and re-tuned for each later imager, so there is no single canonical set; *`shachen.pipeline.run_dust_rgb()`* picks one from the scene's reader. Computed from *`shachen.io.satellite.load_scene()`* fields so it shares DEBRA's 2-km grid and the *`shachen.render`* path; dust appears pink to magenta over dark blue-green surfaces.

`shachen.pipeline.run_dust_rgb()` is the entry point that runs this over a scene, the way *`shachen.pipeline.run_debra()`* runs DEBRA.

Where the recipe comes from (the docs page carries these as links):

- Lensky, I. M., and D. Rosenfeld (2008): Clouds-Aerosols-Precipitation Satellite Analysis Tool (CAPSAT). Atmos. Chem. Phys., 8, 6739-6753, doi:10.5194/acp-8-6739-2008 – the SEVIRI RGB suite this scheme comes from.
- EUMeTrain, "Compilation of RGB Recipes" – the SEVIRI Dust RGB as formalised for operations; the source of *`shachen.constants.DUST_RGB`*.
- NOAA/NASA GOES-R "Quick Guide: Dust RGB" (contributor K. Fuell, NASA SPoRT; CIRA/RAMMB) – the ABI band mix used throughout, and the source of *`shachen.constants.DUST_RGB_ABI`*.

[75] https://doi.org/10.5194/acp-8-6739-2008
[76] https://eumetrain.org/sites/default/files/2020-05/RGB_recipes.pdf
[77] https://rammb.cira.colostate.edu/training/visit/quick_guides/Dust_RGB_Quick_Guide.pdf
[78] https://doi.org/10.1175/JTECH-D-17-0047.1

- Berndt, E., N. Elmer, L. Schultz, and A. Molthan (2018): A Methodology to Determine Recipe Adjustments for Multispectral Composites Derived from Next-Generation Advanced Satellite Imagers. J. Atmos. Oceanic Technol., 35, 643-664, doi:10.1175/JTECH-D-17-0047.1 – why an ABI recipe needs stretches different from SEVIRI's.

shachen.dustrgb.**DUST_RGB_BANDS: tuple**[79]**[*Band*, ...] = (Band.TIR_86, Band.TIR_104, Band.TIR_112, Band.TIR_123)**

The four bands the recipe reads (11.2 um is the non-DEBRA extra).

shachen.dustrgb.**dust_rgb**(*scene: Dataset*[80], *constants:* DustRGBConstants = *DustRGBConstants(red=Bounds(min=-4.0, max=2.0), green=Bounds(min=0.0, max=15.0), green_gamma=2.5, blue=Bounds(min=261.0, max=289.0))*) → DataArray[81]

The classic Dust RGB composite of `scene`, floats in [0, 1].

`scene` needs `bt_tir_86/104/112/123` (K); a missing variable raises ValueError. Returns dims `(y, x, gun)` with coordinate `gun = ["r", "g", "b"]` — the same layout as *`shachen.imagery.enhanced_rgb()`*, so *`shachen.imagery.to_uint8()`* and *`shachen.render.render_debra_png()`* apply unchanged. NaN propagates.

Grids and land mask

Regridding ancillary fields onto the satellite grid. Grid helpers: regrid lat/lon ancillary fields to the satellite grid, land mask.

MERRA-2 skin temperature (0.5 x 0.625 deg) and the CAMEL emissivity climatology (0.05 deg) live on regular lat/lon grids; every DEBRA computation happens on the sensor's 2-km fixed grid, so ancillary data is interpolated onto the scene's pyresample area (never the reverse). The land mask feeds DT3's land/ocean surface shift (Eq. 15).

shachen.geo.**regrid_latlon**(*source: DataArray*[82] | *Dataset*[83], *area: AreaDefinition*[84]) → DataArray[85] | Dataset[86]

Bilinearly interpolate a regular lat/lon-gridded field onto `area`.

`source` must be dimensioned `(lat, lon)` or `(latitude, longitude)` with 1-D coordinate arrays (MERRA-2 and CAMEL conventions, respectively). Returns the field on dims `(y, x)` matching the area's shape; target pixels outside the source extent or off the Earth disk become NaN.

shachen.geo.**land_mask**(*area: AreaDefinition*[87]) → DataArray[88]

Boolean land mask (True = land) on `area`'s grid, dims `(y, x)`.

Backed by the global_land_mask lookup table. Off-disk pixels are False; their value never matters because every BT there is NaN and the confidence factor propagates NaN.

[79] https://docs.python.org/3/library/stdtypes.html#tuple
[80] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[81] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[82] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[83] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[84] https://pyresample.readthedocs.io/en/stable/api/pyresample.geometry.html#pyresample.geometry.AreaDefinition
[85] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[86] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[87] https://pyresample.readthedocs.io/en/stable/api/pyresample.geometry.html#pyresample.geometry.AreaDefinition
[88] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray

Solar geometry

Per-pixel solar zenith angle for the terminator blends. Per-pixel solar zenith angle for the day/terminator/night blends.

Feeds the confidence-factor blending weights (Eqs. 20-21) and the baseline-image blend (Eq. 25). Implemented with pyorbital's astronomy routines on the area's lon/lat arrays.

`shachen.solar.`**`solar_zenith`**(*area:* *AreaDefinition*[89], *when:* *datetime*[90]) → DataArray[91]

Solar zenith angle in degrees on `area`'s grid, dims `(y, x)`.

`when` is the (naive UTC) observation time; the scene start time is accurate enough for the 30-deg-wide terminator blend. Off-disk pixels (non-finite lon/lat) are NaN.

Normalization

The Eq. 3 primitive shared by every test. The Eq. 3 normalization primitive used by every DEBRA test.

`shachen.norm.`**`normalize`**(*x*, *bounds:* Bounds)

N(x) = clip((x - MIN) / (MAX - MIN), 0, 1) (Miller et al. 2017, Eq. 3).

Works on scalars, numpy arrays, and xarray DataArrays. `bounds.min` may exceed `bounds.max` (used for the cos-zenith blends specified in zenith-angle space): the sense of the ramp reverses.

`shachen.norm.`**`normalize_cos_zenith`**(*zenith_deg*, *zenith_bounds_deg:* Bounds, *exponent:* *float*[92])

N(cos(theta); cos(MIN_deg), cos(MAX_deg)) ** exponent (Eqs. 20-21, 25).

Constants

Every tuning bound, offset and weight in one place. All frozen numbers in one place: DEBRA's tuning constants and the classic Dust RGB recipe.

Every (MIN, MAX) scaling bound, offset, and weight from Miller et al. (2017), JGR Atmospheres, doi:10.1002/2017JD027365, using the 26 Feb 2020 erratum versions of Eqs. 7, 21, 22, 24, 25. The paper notes "minor retuning" may be needed per sensor; retune here, nowhere else.

Sign/orientation conventions follow the printed equations exactly.

The sensor facts here (*Band* and the channel maps) are not from the paper; they are what each instrument calls the roles the algorithms read.

class `shachen.constants.`**`Bounds`**

Bases: `object`[93]

(MIN, MAX) pair for the Eq. 3 normalization primitive.

`min:` **`float`**[94]

`max:` **`float`**[95]

`__init__`(*min:* *float*[96], *max:* *float*[97]) → None[98]

[89] https://pyresample.readthedocs.io/en/stable/api/pyresample.geometry.html#pyresample.geometry.AreaDefinition
[90] https://docs.python.org/3/library/datetime.html#datetime.datetime
[91] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray
[92] https://docs.python.org/3/library/functions.html#float

class shachen.constants.**Band**

Bases: StrEnum[99]

Spectral roles the algorithms read (nominal wavelengths in um).

Seven of these are DEBRA's inputs (*DEBRA_BANDS*); TIR_112 exists only for the classic Dust RGB green gun.

VIS_064 = 'vis_064'

cloud mask + day baseline image

NIR_160 = 'nir_160'

daytime cloud test (reserved)

SWIR_39 = 'swir_39'

night thin-cirrus test CM4

WV_62 = 'wv_62'

deep-convection test CM2

TIR_86 = 'tir_86'

dust test DT2 (8.4-8.6 um)

TIR_104 = 'tir_104'

clean window reference (10.3-10.4 um)

TIR_112 = 'tir_112'

classic Dust RGB green gun only (11.2 um; not a DEBRA input)

TIR_123 = 'tir_123'

dirty window, RSW / DT1 (12.3 um)

__new__(*value*)

shachen.constants.**ABI_BANDS: dict[100][*Band*, str[101]] = {Band.NIR_160: 'C05', Band.SWIR_39: 'C07', Band.TIR_104: 'C13', Band.TIR_112: 'C14', Band.TIR_123: 'C15', Band.TIR_86: 'C11', Band.VIS_064: 'C02', Band.WV_62: 'C08'}**

Spectral role -> GOES-R ABI channel name (satpy dataset names).

shachen.constants.**AHI_BANDS: dict[102][*Band*, str[103]] = {Band.NIR_160: 'B05', Band.SWIR_39: 'B07', Band.TIR_104: 'B13', Band.TIR_112: 'B14', Band.TIR_123: 'B15', Band.TIR_86: 'B11', Band.VIS_064: 'B03', Band.WV_62: 'B08'}**

Spectral role -> Himawari AHI band name (satpy dataset names).

[93] https://docs.python.org/3/library/functions.html#object
[94] https://docs.python.org/3/library/functions.html#float
[95] https://docs.python.org/3/library/functions.html#float
[96] https://docs.python.org/3/library/functions.html#float
[97] https://docs.python.org/3/library/functions.html#float
[98] https://docs.python.org/3/library/constants.html#None
[99] https://docs.python.org/3/library/enum.html#enum.StrEnum
[100] https://docs.python.org/3/library/stdtypes.html#dict
[101] https://docs.python.org/3/library/stdtypes.html#str
[102] https://docs.python.org/3/library/stdtypes.html#dict
[103] https://docs.python.org/3/library/stdtypes.html#str

shachen.constants.**DEBRA_BANDS: tuple[104][*Band*, ...] = (Band.VIS_064, Band.NIR_160, Band.SWIR_39, Band.WV_62, Band.TIR_86, Band.TIR_104, Band.TIR_123)**

The seven DEBRA algorithm input roles (Miller et al. 2017 Table 1). `Band.TIR_112` exists only for the classic Dust RGB comparison baseline (*shachen.dustrgb*); pipelines that feed DEBRA alone load these.

shachen.constants.**BAND_CENTER_UM: dict[105][*Band*, float[106]] = {Band.NIR_160: 1.61, Band.SWIR_39: 3.9, Band.TIR_104: 10.33, Band.TIR_112: 11.19, Band.TIR_123: 12.3, Band.TIR_86: 8.44, Band.VIS_064: 0.64, Band.WV_62: 6.19}**

Nominal central wavelengths (um) used to interpolate the UWBF emissivity hinge points to sensor band centers (ABI values; AHI within tolerance).

class shachen.constants.**CloudMaskConstants**

Bases: object[107]

Eqs. 1-12.

cm1_cold_offset_k: float[108] = 50.0

CM1 (Eq. 1): 1 - N(BT10.4; T_skin - cm1_cold_offset_k, T_skin)

cm2: *Bounds*

CM2 (Eq. 4): N(BT10.4 - BT6.2)

cm3: *Bounds*

CM3 (Eq. 5): N(BT10.4 - BT12.3), day/night thin cirrus

cm4: *Bounds*

CM4 (Eq. 6): N(BT3.9 - BT10.4), night-only thin cirrus

r1: *Bounds*

R1 (Eq. 7, erratum): N(BT12.3 - BT10.4) * (1 - CM1)

r2: *Bounds*

R2 (Eqs. 8-9): N(BT8.6 - BT10.4) * restoral weights

cm_norm: *Bounds*

Eq. 12: CM_norm = N(CM)

__init__(*cm1_cold_offset_k: float[109] = 50.0*, *cm2:* Bounds = <*factory*>, *cm3:* Bounds = <*factory*>, *cm4:* Bounds = <*factory*>, *r1:* Bounds = <*factory*>, *r2:* Bounds = <*factory*>, *cm_norm:* Bounds = <*factory*>) → None[110]

class shachen.constants.**DustTestConstants**

Bases: object[111]

Eqs. 13-15.

dt1_max_rsw_k: float[112] = 3.5

DT1 (Eq. 13): (RSW_obs - RSW_bg) / (max_rsw_k - RSW_bg), RSW = BT12.3 - BT10.4

[104] https://docs.python.org/3/library/stdtypes.html#tuple
[105] https://docs.python.org/3/library/stdtypes.html#dict
[106] https://docs.python.org/3/library/functions.html#float
[107] https://docs.python.org/3/library/functions.html#object
[108] https://docs.python.org/3/library/functions.html#float
[109] https://docs.python.org/3/library/functions.html#float
[110] https://docs.python.org/3/library/constants.html#None

dt2_max_btd_k: float[113] = 3.0

DT2 (Eq. 14): (BTD_obs - BTD_bg) / (max_btd_k - BTD_bg), BTD = BT8.6 - BT10.4

dt3_shift_land_k: float[114] = -10.0

DT3 (Eq. 15): (BT10.4 - (T_merra - S - dt3_depth_k)) / dt3_depth_k

dt3_shift_ocean_k: float[115] = 5.0

dt3_depth_k: float[116] = 50.0

__init__(*dt1_max_rsw_k: float[117] = 3.5, dt2_max_btd_k: float[118] = 3.0, dt3_shift_land_k: float[119] = -10.0, dt3_shift_ocean_k: float[120] = 5.0, dt3_depth_k: float[121] = 50.0*) → None[122]

class shachen.constants.**ConfidenceConstants**

Bases: object[123]

Eqs. 16-22 (Eqs. 21-22 per erratum).

dt3_weight_trm: float[124] = 0.5

Eq. 17: CF_trm weights DT3 by this factor; Eq. 18: CF_ngt likewise.

dt3_weight_ngt: float[125] = 0.5

cf_norm: *Bounds*

Eq. 19: each CF variant normalized with these bounds.

blend_exponent: float[126] = 1.5

Eqs. 20-21: terminator blending on cos(theta_sun), exponent 1.5.

ngt_trm_zenith_deg: *Bounds*

night/terminator interface: N(cos theta; cos 105 deg, cos 90 deg)

trm_day_zenith_deg: *Bounds*

terminator/day interface: N(cos theta; cos 90 deg, cos 75 deg)

__init__(*dt3_weight_trm: float[127] = 0.5, dt3_weight_ngt: float[128] = 0.5, cf_norm:* Bounds = <*factory*>, *blend_exponent: float[129] = 1.5, ngt_trm_zenith_deg:* Bounds = <*factory*>, *trm_day_zenith_deg:* Bounds = <*factory*>) → None[130]

[111] https://docs.python.org/3/library/functions.html#object
[112] https://docs.python.org/3/library/functions.html#float
[113] https://docs.python.org/3/library/functions.html#float
[114] https://docs.python.org/3/library/functions.html#float
[115] https://docs.python.org/3/library/functions.html#float
[116] https://docs.python.org/3/library/functions.html#float
[117] https://docs.python.org/3/library/functions.html#float
[118] https://docs.python.org/3/library/functions.html#float
[119] https://docs.python.org/3/library/functions.html#float
[120] https://docs.python.org/3/library/functions.html#float
[121] https://docs.python.org/3/library/functions.html#float
[122] https://docs.python.org/3/library/constants.html#None

class shachen.constants.**ImageryConstants**

Bases: object[131]

Eqs. 23-29 (Eqs. 24-25 per erratum).

bg_blend_zenith_deg: ***Bounds***

Eq. 25 (erratum): B_bg = 1 - N(theta_sun; 79 deg, 89 deg)^1.5

bg_blend_exponent: **float**[132] **= 1.5**

cf_cap: **float**[133] **= 0.5**

Eqs. 27-29: RED = GRN = BI*(1 - min(CF, cf_cap)) + CF; BLU = BI*(1 - min(CF, cf_cap)) + blue_dimming * CF

blue_dimming: **float**[134] **= 0.1**

gun_max: **float**[135] **= 1.2**

Per-gun rescale [0, gun_max] -> [0, 255]

__init__(*bg_blend_zenith_deg:* Bounds = <*factory*>, *bg_blend_exponent:* *float*[136] = *1.5*, *cf_cap:* *float*[137] = *0.5*, *blue_dimming:* *float*[138] = *0.1*, *gun_max:* *float*[139] = *1.2*) → None[140]

shachen.constants.**COLOR_DIMMING: dict**[141]**[str**[142]**, tuple**[143]**[float**[144]**, float**[145]**, float**[146]**]] = {'blue': (0.25, 0.25, 1.0), 'green': (0.1, 1.0, 0.1), 'pink': (1.0, 0.25, 0.25), 'yellow': (1.0, 1.0, 0.1)}**

Per-gun dimming triples (D_red, D_grn, D_blu) generalizing Eqs. 27-29: each gun is BI*(1 - min(CF, cf_cap)) + D_gun*CF, a full gun has D = 1.0. Presets from the paper's section 4.2 text ("dust can be made pink by applying D = 0.25 to equations (28) and (29), green by applying D = 0.10 to equations (27) and (29), or blue by applying D = 0.25 to (27) and (28)").

shachen.constants.**COMPOSITE_WINDOW_DAYS: int**[147] **= 14**

Cloud-cleared composite background (paper section 3.2, Option B): rolling window length ("multiday (e.g., 14 day) composites") and the fewest composite days accepted at load time before the background is refused.

[123] https://docs.python.org/3/library/functions.html#object
[124] https://docs.python.org/3/library/functions.html#float
[125] https://docs.python.org/3/library/functions.html#float
[126] https://docs.python.org/3/library/functions.html#float
[127] https://docs.python.org/3/library/functions.html#float
[128] https://docs.python.org/3/library/functions.html#float
[129] https://docs.python.org/3/library/functions.html#float
[130] https://docs.python.org/3/library/constants.html#None
[131] https://docs.python.org/3/library/functions.html#object
[132] https://docs.python.org/3/library/functions.html#float
[133] https://docs.python.org/3/library/functions.html#float
[134] https://docs.python.org/3/library/functions.html#float
[135] https://docs.python.org/3/library/functions.html#float
[136] https://docs.python.org/3/library/functions.html#float
[137] https://docs.python.org/3/library/functions.html#float
[138] https://docs.python.org/3/library/functions.html#float
[139] https://docs.python.org/3/library/functions.html#float
[140] https://docs.python.org/3/library/constants.html#None
[141] https://docs.python.org/3/library/stdtypes.html#dict
[142] https://docs.python.org/3/library/stdtypes.html#str
[143] https://docs.python.org/3/library/stdtypes.html#tuple
[144] https://docs.python.org/3/library/functions.html#float
[145] https://docs.python.org/3/library/functions.html#float
[146] https://docs.python.org/3/library/functions.html#float

class shachen.constants.**DustRGBConstants**

Bases: `object`[148]

One classic Dust RGB stretch set. Not part of DEBRA (Eqs. 1-29).

Each gun is `N(x; MIN, MAX) ** (1/gamma)` over the GOES-R/Himawari band mix of the CIRA Dust RGB Quick Guide (satpy's `dust` composite): 12.3-10.3, 11.2-8.4, 10.3 — note the green minuend is 11.2 um, not the 10.8 um of SEVIRI's `IR10.8 - IR8.7`.

The stretch values are **per sensor**: *`DUST_RGB`* (SEVIRI) and *`DUST_RGB_ABI`* are the two published sets, dispatched by *`DUST_RGB_BY_READER`*. See *`shachen.dustrgb`* for the references.

red: ***Bounds***

RED: BT12.3 - BT10.4 (split window), [-4, +2] K, gamma 1.

green: ***Bounds***

GRN: BT11.2 - BT8.6, [0, 15] K, gamma 2.5 (note: 11.2 um, not 10.4).

green_gamma: **float**[149] **= 2.5**

blue: ***Bounds***

BLU: BT10.4, [261, 289] K, gamma 1.

__init__(*red:* Bounds = <*factory*>, *green:* Bounds = <*factory*>, *green_gamma:* *float*[150] = *2.5*, *blue:* Bounds = <*factory*>) → None[151]

shachen.constants.**DUST_RGB = DustRGBConstants(red=Bounds(min=-4.0, max=2.0), green=Bounds(min=0.0, max=15.0), green_gamma=2.5, blue=Bounds(min=261.0, max=289.0))**

The original SEVIRI stretches (Lensky and Rosenfeld 2008, as formalised in EUMeTrain's recipe compilation; satpy's generic `dust_default` enhancement). Applied to Himawari here, which is what satpy does too: it ships no `dust_ahi` override. Also the fallback for an unknown sensor.

shachen.constants.**DUST_RGB_ABI = DustRGBConstants(red=Bounds(min=-6.7, max=2.6), green=Bounds(min=-0.5, max=20.0), green_gamma=2.5, blue=Bounds(min=261.2, max=288.7))**

The ABI-adjusted stretches from the CIRA GOES-R Dust RGB Quick Guide (satpy's sensor-specific `dust_abi` enhancement), converted from the Quick Guide's degrees Celsius. Berndt et al. (2018) is the method behind the adjustment: ABI's channels do not sit where SEVIRI's do, so the SEVIRI numbers put the clear-sky land background at the wrong end of the red stretch.

shachen.constants.**DUST_RGB_BY_READER: dict**[152]**[str**[153]**, *DustRGBConstants*] = {'abi_l1b': DustRGBConstants(red=Bounds(min=-6.7, max=2.6), green=Bounds(min=-0.5, max=20.0), green_gamma=2.5, blue=Bounds(min=261.2, max=288.7)), 'ahi_hsd': DustRGBConstants(red=Bounds(min=-4.0, max=2.0), green=Bounds(min=0.0, max=15.0), green_gamma=2.5, blue=Bounds(min=261.0, max=289.0))}**

satpy reader -> the stretch set that sensor's operational product uses. *`shachen.pipeline.run_dust_rgb()`* reads this off `scene.attrs["reader"]` so a baseline rendered here matches the image forecasters see. The same rule is why there is no single canonical recipe to freeze: the

[147] https://docs.python.org/3/library/functions.html#int
[148] https://docs.python.org/3/library/functions.html#object
[149] https://docs.python.org/3/library/functions.html#float
[150] https://docs.python.org/3/library/functions.html#float
[151] https://docs.python.org/3/library/constants.html#None

scheme is retuned per imager (satpy carries `ash_abi`, `convection_abi` and the night-microphysics variants for the same reason).

class shachen.constants.**DebraConstants**

Bases: `object`[154]

The whole tuning surface, one group per stage of the algorithm.

cloud_mask: ***CloudMaskConstants***

dust_tests: ***DustTestConstants***

confidence: ***ConfidenceConstants***

imagery: ***ImageryConstants***

__init__(*cloud_mask:* CloudMaskConstants = <*factory*>, *dust_tests:* DustTestConstants = <*factory*>, *confidence:* ConfidenceConstants = <*factory*>, *imagery:* ImageryConstants = <*factory*>) → None[155]

shachen.constants.**DEFAULTS** =
DebraConstants(cloud_mask=CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8)), dust_tests=DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0), confidence=ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.25, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0)), imagery=ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2))

The paper-tuned default constant set.

shachen.constants.**ABI_TUNED** =
DebraConstants(cloud_mask=CloudMaskConstants(cm1_cold_offset_k=50.0, cm2=Bounds(min=0.0, max=25.0), cm3=Bounds(min=2.0, max=4.5), cm4=Bounds(min=5.0, max=8.0), r1=Bounds(min=0.0, max=3.5), r2=Bounds(min=-1.0, max=3.0), cm_norm=Bounds(min=0.45, max=0.8)), dust_tests=DustTestConstants(dt1_max_rsw_k=3.5, dt2_max_btd_k=3.0, dt3_shift_land_k=-10.0, dt3_shift_ocean_k=5.0, dt3_depth_k=50.0), confidence=ConfidenceConstants(dt3_weight_trm=0.5, dt3_weight_ngt=0.5, cf_norm=Bounds(min=0.4, max=2.5), blend_exponent=1.5, ngt_trm_zenith_deg=Bounds(min=105.0, max=90.0), trm_day_zenith_deg=Bounds(min=90.0, max=75.0)), imagery=ImageryConstants(bg_blend_zenith_deg=Bounds(min=79.0, max=89.0), bg_blend_exponent=1.5, cf_cap=0.5, blue_dimming=0.1, gun_max=1.2))

Optional retune for the ABI + MERRA-2 + CAMEL stack. Single deviation from the paper: the Eq. 19 lower bound is raised 0.25 -> 0.40. Rationale: on this ancillary stack DT3 carries a 0.2-0.5

[152] https://docs.python.org/3/library/stdtypes.html#dict
[153] https://docs.python.org/3/library/stdtypes.html#str
[154] https://docs.python.org/3/library/functions.html#object
[155] https://docs.python.org/3/library/constants.html#None

clear-sky floor over land (MERRA-2 skin T runs warmer than BT10.4, plus the -10 K land shift), which leaks through low-cloud-mask pixels as a faint yellow tint over vegetated and low-cloud areas. Sweeping the bound over the three reference cases: 0.40 removes 92% of the tinted area (SE-US box, fraction CF > 0.05: 0.37 -> 0.028) while the 2017-03-23 plume mean CF drops only 5% (0.476 -> 0.452) and the 2020-12-23 plume core stays distinct (p90 0.40). The paper itself anticipates "minor retuning" per sensor.

I/O

Readers for the three inputs DEBRA needs: calibrated L1b imagery, MERRA-2 skin temperature, and the surface emissivity climatology. None of this is on the `import shachen` path — each function needs the `satellite` or `data` extra.

Satellite imagery

Satellite L1b -> calibrated BT/reflectance fields on the 2-km IR grid.

satpy does the radiance -> BT / reflectance calibration and the native down-sampling of the VIS/NIR bands onto the coarsest (2 km) grid, so ABI netCDF and Himawari AHI HSD share one code path.

`shachen.io.satellite.`**`load_scene`**(*files: Iterable*[156]*[str*[157] *| Path*[158]*], reader: str*[159] *= 'abi_l1b', roles: Iterable*[160]*[*Band*] | None*[161] *= None, bbox: tuple*[162]*[float*[163]*, float*[164]*, float*[165]*, float*[166]*] | None*[167] *= None*) → Dataset[168]

Load L1b files and return a Dataset on the sensor's 2-km fixed grid.

Variables are named by DEBRA role: `bt_tir_104` etc. for emissive bands (K), `refl_vis_064` etc. for reflective bands (%). The pyresample AreaDefinition is attached as `ds.attrs["area"]` and the scan start time as `ds.attrs["start_time"]`.

`roles` restricts loading to a subset of DEBRA bands (e.g. `shachen.composite.COMPOSITE_BANDS` for TIR-only composite days, whose directories hold only those bands' files); `None` loads all seven.

`bbox = (lon_min, lat_min, lon_max, lat_max)` in degrees crops the scene (satpy `Scene.crop(ll_bbox=...)`) *after* the native resample, so every variable and `attrs["area"]` describe the cropped grid — This trims AHI full disks (5500 x 5500 at 2 km) to the domain of interest. `None` keeps the full scene.

[156] https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterable
[157] https://docs.python.org/3/library/stdtypes.html#str
[158] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[159] https://docs.python.org/3/library/stdtypes.html#str
[160] https://docs.python.org/3/library/collections.abc.html#collections.abc.Iterable
[161] https://docs.python.org/3/library/constants.html#None
[162] https://docs.python.org/3/library/stdtypes.html#tuple
[163] https://docs.python.org/3/library/functions.html#float
[164] https://docs.python.org/3/library/functions.html#float
[165] https://docs.python.org/3/library/functions.html#float
[166] https://docs.python.org/3/library/functions.html#float
[167] https://docs.python.org/3/library/constants.html#None
[168] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset

MERRA-2 reanalysis

MERRA-2 hourly surface skin temperature (TS from M2T1NXSLV).

Downloads via earthaccess using the Earthdata login in `~/.netrc`. The full tavg1_2d_slv_Nx day file is large, so it is stripped after download to a small TS-only netCDF cache, which the pipeline regrids onto the satellite grid.

`shachen.io.merra.`**`SURFACE_MET_SLV_VARS = ('T2M', 'QV2M', 'PS', 'U10M', 'V10M')`**

Single-level variables the dust-PM10 matchup keeps from M2T1NXSLV.

`shachen.io.merra.`**`FLX_SHORT_NAME = 'M2T1NXFLX'`**

Boundary-layer height comes from the surface-flux collection.

`shachen.io.merra.`**`fetch_skin_temperature`**(*day:* *date*[169], *out_dir:* *Path*[170]) → Path[171]

Download MERRA-2 TS for `day`; return path to a TS-only netCDF.

`shachen.io.merra.`**`fetch_surface_meteorology`**(*day:* *date*[172], *out_dir:* *Path*[173]) → Path[174]

Download the dust-PM10 met covariates for `day`; return a cache path.

The cache `merra2_met_<%Y%m%d>.nc` merges `SURFACE_MET_SLV_VARS` from M2T1NXSLV with `SURFACE_MET_FLX_VARS` (PBLH) from M2T1NXFLX, both hourly on the native 0.5 x 0.625 degree grid.

`shachen.io.merra.`**`load_surface_meteorology`**(*path:* *Path*[175], *when:* *datetime*[176]) → Dataset[177]

Load the met covariates interpolated to `when`.

Same half-past-the-hour time stamps, and the same clamping at the ends of the day, as *`load_skin_temperature()`*.

`shachen.io.merra.`**`load_skin_temperature`**(*path:* *Path*[178], *when:* *datetime*[179]) → DataArray[180]

Load TS (K) interpolated to `when` on the MERRA-2 grid.

Linear in time between the half-past-the-hour stamps, clamped to the file's first and last hourly mean — see `_interp_to_time()`.

Surface emissivity

IR surface emissivity climatology, interpolated to sensor band centers.

The paper uses the monthly UW Baseline Fit (UWBF, CIMSS) 0.05-degree climatology, which needs a separate CIMSS registration. The default here is its successor, CAMEL (CAM5K30EM on NASA LP DAAC), reachable with the same Earthdata credentials as MERRA-2. A downloaded UWBF file works too: both formats carry an emissivity cube on labelled hinge-point wavelengths, and interpolation is linear in wavelength, as the paper implies.

[169] https://docs.python.org/3/library/datetime.html#datetime.date
[170] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[171] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[172] https://docs.python.org/3/library/datetime.html#datetime.date
[173] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[174] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[175] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[176] https://docs.python.org/3/library/datetime.html#datetime.datetime
[177] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[178] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[179] https://docs.python.org/3/library/datetime.html#datetime.datetime
[180] https://docs.xarray.dev/en/stable/generated/xarray.DataArray.html#xarray.DataArray

shachen.io.emissivity.**EMISSIVE_BANDS = (Band.SWIR_39, Band.WV_62, Band.TIR_86, Band.TIR_104, Band.TIR_123)**

DEBRA only needs emissivity for the emissive bands used by the background.

shachen.io.emissivity.**fetch_emissivity**(*month: date*[181], *out_dir: Path*[182]) → Path[183]

Download the CAMEL monthly emissivity file covering `month`.

shachen.io.emissivity.**load_band_emissivity**(*path: Path*[184], *bands: tuple*[185]*[*Band, *...] = (Band.SWIR_39, Band.WV_62, Band.TIR_86, Band.TIR_104, Band.TIR_123)*) → Dataset[186]

Load hinge-point emissivity and interpolate to DEBRA band centers.

Returns a Dataset with one `emis_<band>` variable per requested band on the climatology's native lat/lon grid, values masked where the file has no retrieval (ocean/fill).

2.4 Deviations from the published algorithm

Everything here is deliberate, unit-tested, and recorded in the module docstrings next to the code. Check this page and the erratum before changing any of it.

The baseline is Miller et al. (2017), doi:10.1002/2017JD027365[187], as amended by the erratum published 26 February 2020.

2.4.1 1. Equations amended by the erratum

Equations 7, 21–22 and 24–25 are implemented in their erratum form. If you are reading the 2017 PDF, these five will not match the code; the difference comes from the erratum rather than from this implementation.

2.4.2 2. Corrections to equations the erratum does not cover

Three printed equations remain inconsistent with the paper's own prose and figures after the erratum. Each is implemented per the prose, with the reasoning below.

Eq. 4 (CM2) — magnitude reversed

```
CM2 = 1 - N(BT10.4 - BT6.2; 0, 25)      # deep convection → 1, clear → 0
```

As printed, clear sky — where the 10.4/6.2 μm split runs 25–55 K — yields CM2 $\approx$ 1, which saturates the cloud mask everywhere and degenerates the whole algorithm. The prose describes CM2 as behaving "in a similar fashion" to the already-reversed Eq. 1, and Figure 3c shows the reversed sense.

[181] https://docs.python.org/3/library/datetime.html#datetime.date
[182] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[183] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[184] https://docs.python.org/3/library/pathlib.html#pathlib.Path
[185] https://docs.python.org/3/library/stdtypes.html#tuple
[186] https://docs.xarray.dev/en/stable/generated/xarray.Dataset.html#xarray.Dataset
[187] https://doi.org/10.1002/2017JD027365

Eq. 11 (CM_day) — CM3 in place of CM4

```
CM_day = (CM1 + CM2 + CM3) * (1 - max(R1, R2_day))
```

The printed equation references CM4, but CM4 is the 3.9 μm test, which the paper itself defines as night-only. CM3 is the intended term.

Eq. 15 (DT3) — magnitude reversed

```
DT3 = clip(((T_MERRA - S) - BT10.4) / depth, 0, 1)
```

with the surface shift `S` = −10 K (land) / +5 K (ocean) applied as printed, and `depth` = 50 K. The prose is explicit that "observations that are relatively cold compared to MERRA produce high value for DT3", which the printed equation does not do.

2.4.3 3. Ancillary data substitution

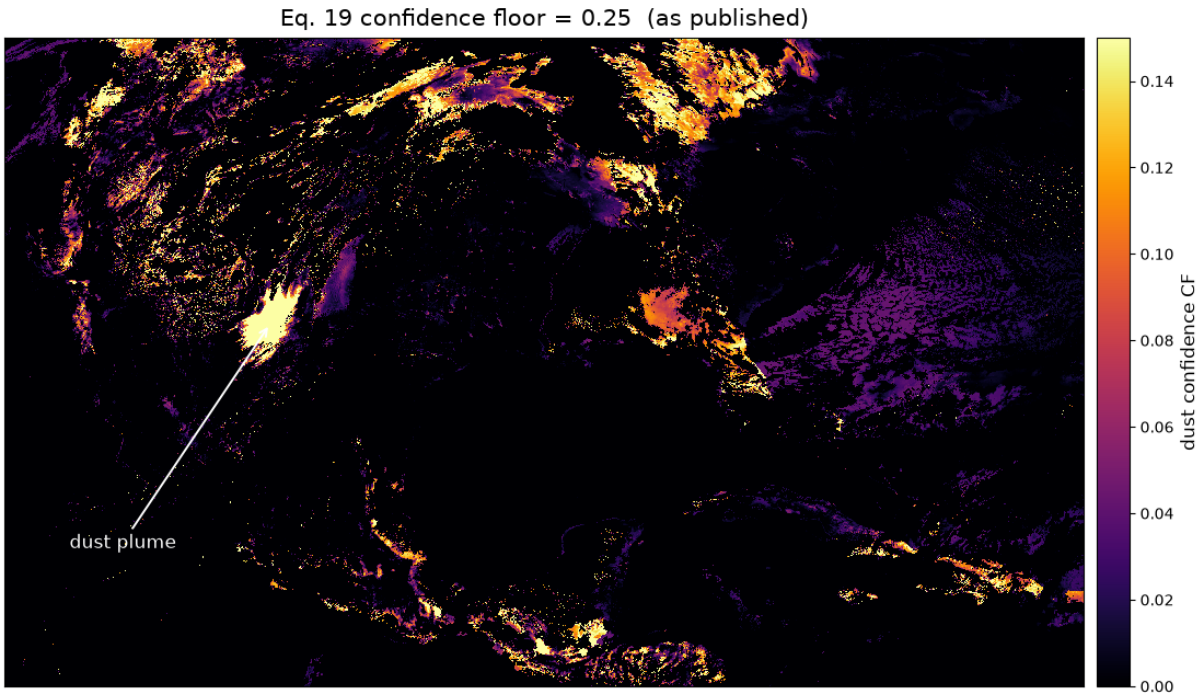
Surface emissivity: CAMEL in place of UWBF. The paper specifies the monthly UW Baseline Fit (UWBF, CIMSS) 0.05° climatology, which requires a separate CIMSS registration. The default here is its successor CAMEL (`CAM5K30EM`, NASA LP DAAC), reachable with the same Earthdata credentials as MERRA-2. Both carry an emissivity cube on labelled hinge-point wavelengths and are interpolated linearly in wavelength, as the paper implies; `io.emissivity.load_band_emissivity` accepts either file.

2.4.4 4. Optional ABI retune (not the default)

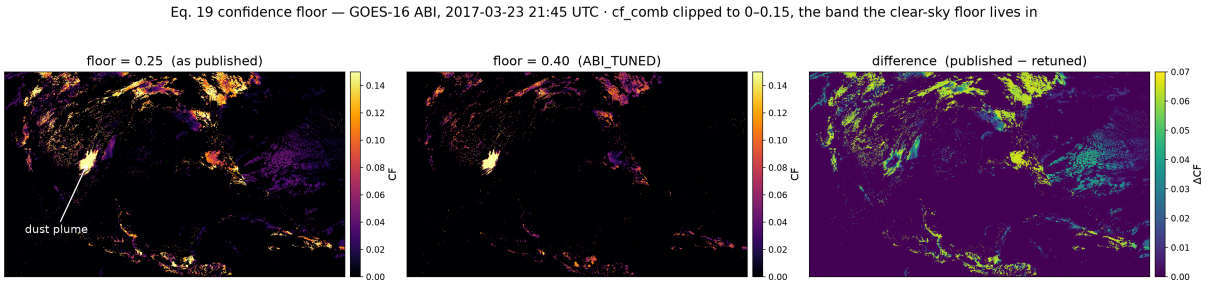
`constants.ABI_TUNED` raises the Eq. 19 lower bound from 0.25 to 0.40. It is opt-in; `DEFAULTS` remains the paper's values.

Why it exists: with this ancillary stack (ABI + MERRA-2 + CAMEL), DT3 carries a 0.2–0.5 clear-sky floor over land, because MERRA-2 skin temperature runs warmer than BT10.4 and the −10 K land shift adds to it. That floor leaks through low-cloud-mask pixels as a faint yellow tint over vegetated and low-cloud areas.

The images below are APNG: they animate in any modern browser and show their first frame everywhere else. The blink presentation is used because the change is a small shift in a low-amplitude field, which side-by-side panels hide.



`cf_comb` clipped to 0–0.15, the band containing the clear-sky floor. The published floor leaks a speckle of confidence across the whole scene and a haze over the vegetated southeast; the retune removes most of it. The dust plume is untouched.



The same two states with their difference. ΔCF never exceeds 0.067, and is spread over a large area.

The same change in the finished imagery, cropped from the plume east to the Gulf coast and magnified 2×. Here the two states differ by at most 14/255 per pixel: the tint being removed sits at CF 0.05–0.10, which Eqs. 27–29 map to a handful of RGB levels. Running both settings therefore produces output images that look nearly identical, while the statistic below changes substantially.

Reproduce either state with `python scripts/run_case.py 2017-03-23-swus --tuning pa-`

`per|abi`.

Sweeping the floor across the three reference cases:

metric	0.25 (published)	0.30	0.35	0.40 (`ABI_TUNED`)	0.45
2017-03-23 plume mean CF	0.476	0.468	0.460	0.452	0.445
2017-03-23 SE-US vegetation, fraction CF > 0.05	0.368	0.340	0.172	0.028	0.006
2020-12-23 plume core mean	0.202	0.184	0.166	0.150	0.135
2020-12-23 plume core p90	0.439	0.426	0.413	0.399	0.384

0.40 is the knee: 92% of the tinted area is gone for a 5% cost on the strong 2017 plume, and the moderate 2020-12-23 core still renders as saturated yellow (p90 $\approx$ 0.40 after Eqs. 27–29). 0.45 starts erasing the moderate case.

The paper anticipates "minor retuning" per sensor. Himawari AHI needs no retune: the published bounds were tuned on AHI and transfer as-is. `ABI_TUNED` is selectable on AHI but has not been validated there.

2.4.5 What is not a deviation

- **The composite background** (`composite.py`) is the paper's own §3.2 alternative: a ~14-day rolling, cloud-cleared, same-time-of-day composite. It is offered alongside the semi-analytic background because it carries the split-window water-vapour depression that emissivity $\times$ Planck lacks, which otherwise zeroes DT1/DT2 on transparent winter plumes.
- **Himawari AHI support** is within the paper's scope; the band mapping differs from ABI.
- **`constants.py` values.** Every calibration bound, offset and weight is transcribed from the paper and unit-tested against an independent transcription.

2.5 Patent history

An early formulation of the DEBRA algorithm was patented. The patent expired in 2024; this page records the public record. Development of this implementation began in August 2026, more than two years after the patent lapsed.

Not legal advice

This page is historical documentation of public patent records, last verified 2026-08-29 against the sources linked at the bottom. It is not legal advice.

2.5.1 The patent

Grant	US 9,383,478 B2[188], "System and method for atmospheric parameter enhancement"
Inventor	Steven D. Miller (the paper's first author)
Assignee	The USA as represented by the Secretary of the Navy (reel/frame 037279/0473, executed 2015-12-02)
Application	14/150,467, filed 2014-01-08
Priority	Provisional 61/756,555, 2013-01-25
Granted	2016-07-05
Nominal expiration	2034-09-01
Actual status	Expired 2024-07-05 for non-payment of the 8th-year maintenance fee (37 CFR 1.362)

The USPTO legal-event record: the 4th-year maintenance fee was paid (2019-12-18), the 8th-year reminder was mailed 2024-02-26, no payment followed, the lapse was recorded 2024-08-12 with effect from 2024-07-05. No petition to reinstate appears in the Google Patents legal-event record as of the verification date. Whether one is pending in the USPTO's own transaction history has not been checked; Patent Center requires a login.

2.5.2 No patent outside the United States

The patent's only family member (DOCDB family 51223030) is PCT application WO 2014/116472 A1[189] (PCT/US2014/011586), which ceased without entering the national phase in any country. There has never been a corresponding patent in Japan, China, Europe, or anywhere else. There are also no continuations, divisionals, or continuations-in-part within this family: application 14/150,467 is the only US member of DOCDB family 51223030.

Note

That statement is scoped to the family. A later application by the same inventor — an ABI adaptation, or an ash or fog variant — would not appear in family 51223030 and is not covered here. Establishing that would take an inventor-name search of the USPTO full-text database, which has not been done.

Two predecessor patents by the same inventor and assignee — US 7,242,803 and US 7,379,592, the SeaWiFS-era "significant dust detection and enhancement" algorithms (priority 2003) — expired at the end of their 20-year terms.

[188] https://patents.google.com/patent/US9383478B2/en
[189] https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2014116472

2.5.3 Constants in the patent differ from the published algorithm

The patent recites specific constants for the algorithm's tests. They predate the published paper and differ from what this package (which follows Miller et al. 2017, as amended by the 2020 erratum) implements. Recorded here as provenance for anyone cross-checking where a number came from:

Quantity	Patent claims (2014)	Miller et al. 2017 / this package
Reference window channel	11.2 μm	10.3/10.4 μm
Split-window (BTD1 / DT1) upper bound	4.0 K	3.5 K
8.5 μm test (BTD2 / DT2) upper bound	0.5 K	3.0 K
Confidence normalization	0.5 – 2.5	0.25 – 2.50
Terminator weighting	threshold on cos θ > 0.383	zenith-band blending, exponent 1.5
RGB composition: CF cap / blue dimming / gun max	0.5 / 0.1 / 1.2	0.5 / 0.1 / 1.2

2.5.4 Lineage, 2001–2017

Note

This section is a reconstruction assembled from public documents — the patents and papers cited here. The equations, channel tables, and dates are transcribed from those sources. The sequence, and the reasoning attributed to each change, are inferences drawn from reading them; they are not an account given by the authors. Nothing on this page has been reviewed or confirmed by Steven D. Miller, the Naval Research Laboratory, CIRA, or Colorado State University.

The algorithm predates the 2013 application. Its two predecessors — US 7,242,803 (application 10/713,908, recited on the US 7,379,592 front page as filed 2003-01-21) and its continuation-in-part US 7,379,592 (application 10/885,526, filed 2004-06-30) — describe a product that was already in routine operational use and had already been revised. Automated processing of the SeaWiFS data "commenced Aug. 8, 2001"; the dust case in Fig. 3 was captured at the Navy Regional Center in Rota, Spain, on 2001-02-13 1255Z, one of the three receiving stations named in the text along with Bahrain and Yokosuka.

Everything quoted or transcribed below is from the US 7,379,592 specification unless stated otherwise.

Four stages

Stage	Whe	What changed	Limitation of that stage
Sea-WiFS, visible only	2001 2002	Dust over water read as a colour anomaly	No infrared channels; land not addressable
MODIS, “inclu-sive” logic (Eq. 3)	2003 2004	Thermal infrared added; the four terms summed	“the cost of this aggression is a high frequency of false alarms” — most commonly “cold land, cloud shadows, thin cirrus, and sunglint upon unmasked lake bodies”
“Exclu-sive” logic (Eq. 5)	2004	The same four terms multiplied, so all must agree	“it only takes a single zero-valued term to set the entire $D_{\text{lnd}}^{\text{new}}$ term to zero”, and the split-window term weakens for very thick dust
DEBRA	2013 2017	Weighted maximum; background from external priors; masks become continuous weights; a per-pixel con-fidence factor as the output	—

The limitations quoted for the inclusive and exclusive stages are stated in the specification. Arranging the four into a single line of descent, each stage driven by the failures of the one before it, is an inference drawn from reading them in order; the sources do not present themselves that way.

The two 2004 formulations coexisted: “Equation 5 should not supplant Equation 3, but rather be consid-ered as a variation of the technique providing superior results in certain dust scenarios.” As of 2004 the combination logic was, on the record, still unsettled.

The 2004 land algorithm, as recited

$T(n)$ is the brightness temperature of MODIS channel n in kelvins, $R(n)$ its normalized reflectance, and $T_{\text{max}}(31)$ the maximum pixel temperature in the current scene.

The red gun carries the dust enhancement; the blue and green guns are the same as in the over-water algorithm.

$$D_{\text{lnd}} = L_1 + L_3 - L_4 + (1.0 - L_2) \quad \text{(Eq. 3, inclusive; scaled } [1.3,\ 2.7]\text{)}$$
$$D_{\text{lnd}}^{\text{new}} = L_1\,(1.0 - L_2)\,L_3\,(1.0 - L_4) \quad \text{(Eq. 5, exclusive; scaled } [0.35,\ 0.75]\text{)}$$

The same four terms appear in both (Table 2):

Term	Expression	Normalization bounds
L_1	$T(32) - T(31)$	$-2 \to 2$ K
L_2	$T(31)$	$T_{\text{dyn}}(31) \to T_{\text{max}}(31)$
L_3	$2R(1) - R(3) - R(4) - L_2$	$-1.5 \to 0.25$
L_4	$R(26) > 0.05\ ?\ 0$, else 1	(n/a)

and the dynamic temperature floor that L_2 scales against:

$$T_{\text{dyn}} = \begin{cases} T_{\max}(31) - 21 & \text{if } T_{\max}(31) < 301 \text{ K} \\ \big(T_{\max}(31) - 273\big)/4 + 273 & \text{otherwise} \end{cases} \qquad \text{(Eq. 4)}$$

Channels used (Table 1), with Rayleigh scatter removed from 1–4:

Channel	λ (μm)	Resolution (km)	Description
1	0.645	0.25	Red
2	0.853	0.25	Reflective IR
3	0.469	0.50	Blue
4	0.555	0.50	Green
26	1.38	1.0	Shortwave Vapor
31	11.0	1.0	IR Window 1
32	12.0	1.0	IR Window 2

The over-water branch of the same patent is a normalized difference of two SeaWiFS reflectances α_λ at λ nm,

$$\Delta = \frac{\alpha_{865} - \alpha_{412}}{\alpha_{865} + \alpha_{412}}$$

whose logarithm is "scaled between -0.45 and 0.20 and loaded into the red channel of the RGB composite". Its cloud screen is a mean of the 412, 555 and 670 nm channels exceeding 50% with a standard deviation below 2.5%.

Where "Dynamic" comes from

T_{dyn} is a temperature floor that moves with the scene, computed from the scene's own $T_{\max}(31)$. The stated reason for it is that "the dynamic temperature scaling (Equation 4) was introduced to reduce seasonal and diurnal effects giving rise to false detection over cold land." That is a background estimate that adapts to the scene, the same property that names the "Dynamic" in DEBRA; the sources do not state that connection themselves. Between 2004 and 2013 the background changes source rather than principle: it stops being a statistic of the current scene and becomes an external prior, a surface emissivity database plus a skin-temperature reanalysis, which is what this package implements in `background.py`.

Where the constants come from

On the normalization bounds, the specification says they "were determined experimentally based on a wide variety of dust case studies, with values selected toward optimizing dust contrast while maintaining an enhancement appearance consistent with the over-water algorithm."

Two things follow, both stated rather than inferred. The bounds are empirical, fitted to case studies. And one of the fitting objectives was cross-algorithm visual continuity: the method is "composed of two algorithms (over-land and water) tuned to maintain a similar enhancement of dust crossing coastlines", so that a dust front "maintains a similar appearance across the land/sea algorithmic boundary." The specification also records that "performance testing for the new method is ongoing, with only minor corrections to scaling bounds anticipated": the document describes work in progress.

Two transcription caveats

- The specification contains an apparent slip. Discussing the weakness of Eq. 5 it reads "the Split-Window L_2 terms decreases in strength for very thick dust" [sic], but per its own Table 2 the split window is L_1; L_2 is the 11 μm brightness temperature. The table above follows Table 2.
- The specification refers twice to "the computer program code listing in the Appendix", but the granted text of US 7,379,592 B2 is 18 columns of specification with 2 claims and 14 sheets of drawings, and contains no code listing. The patent as published contains no reference implementation. Whether the appendix was filed can only be settled from the image file wrapper of 10/885,526 or of the parent 10/713,908; that has not been checked.

Sources for this section

- US 7,242,803[190] — "System and method for significant dust detection and enhancement"
- US 7,379,592[191] — the continuation-in-part, source of every equation, table, date and quotation above
- Miller, S. D. (2003), A consolidated technique for enhancing desert dust storms with MODIS, *Geophys. Res. Lett.*, 30(20), 2071, doi:10.1029/2003GL018279[192] — cited on the US 7,379,592 face as the forthcoming GRL paper for the MODIS land/ocean enhancement
- Miller et al. (2017), doi:10.1002/2017JD027365[193], as amended by the erratum of 26 February 2020 — the baseline this package implements, see *Deviations*

Unverified

A correction to the 2003 GRL paper is reported to have been published 2020-06-10. Neither that correction nor its relationship to the 2020 erratum on Miller et al. (2017) has been checked here, and *Deviations* tracks only the latter. Open item.

2.5.5 Verifying the current status

- Legal events and family: Google Patents[194]
- Prosecution and transaction history (free USPTO.gov account required): USPTO Patent Center, application 14/150,467[195]
- Maintenance fees (enter patent 9383478, application 14150467): USPTO fee storefront[196]
- Ceased PCT family member: WIPO PatentScope, WO 2014/116472[197]
- Later applications by the same inventor (not covered above; the name is a common one, so filter by assignee and subject matter): USPTO Patent Public Search[198], inventor name search
- Assignment records:

[190] https://patents.google.com/patent/US7242803B2/en
[191] https://patents.google.com/patent/US7379592B2/en
[192] https://doi.org/10.1029/2003GL018279
[193] https://doi.org/10.1002/2017JD027365
[194] https://patents.google.com/patent/US9383478B2/en
[195] https://patentcenter.uspto.gov/applications/14150467
[196] https://fees.uspto.gov/MaintenanceFees/
[197] https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2014116472
[198] https://ppubs.uspto.gov/pubwebapp/

USPTO Assignment Search[199]

[199] https://assignmentcenter.uspto.gov/search/patent/abstract%3FapplicationNumber%3D14150467

CHAPTER

THREE

CITATION

If you use this software, please cite the original algorithm:

> Miller, S. D., Bankert, R. L., Grasso, L. D., Lindsey, D. T., Kuciauskas, A. P., & Combs, C. L. (2017). A dynamic enhancement with background reduction algorithm: Overview and application to satellite-based dust storm detection. *Journal of Geophysical Research: Atmospheres*, 122, 12,938–12,959. https://doi.org/10.1002/2017JD027365

and, for the implementation, the metadata in `CITATION.cff`.

This is an independent implementation. It is not produced, endorsed, or verified by the papers' authors, by EUMETSAT, by CIRA, or by NOAA.

PYTHON MODULE INDEX

S

INDEX

Symbols

A

B

C

D

E

F

G

I

L

M

N

P

R

S

T

V

W