OroraTech

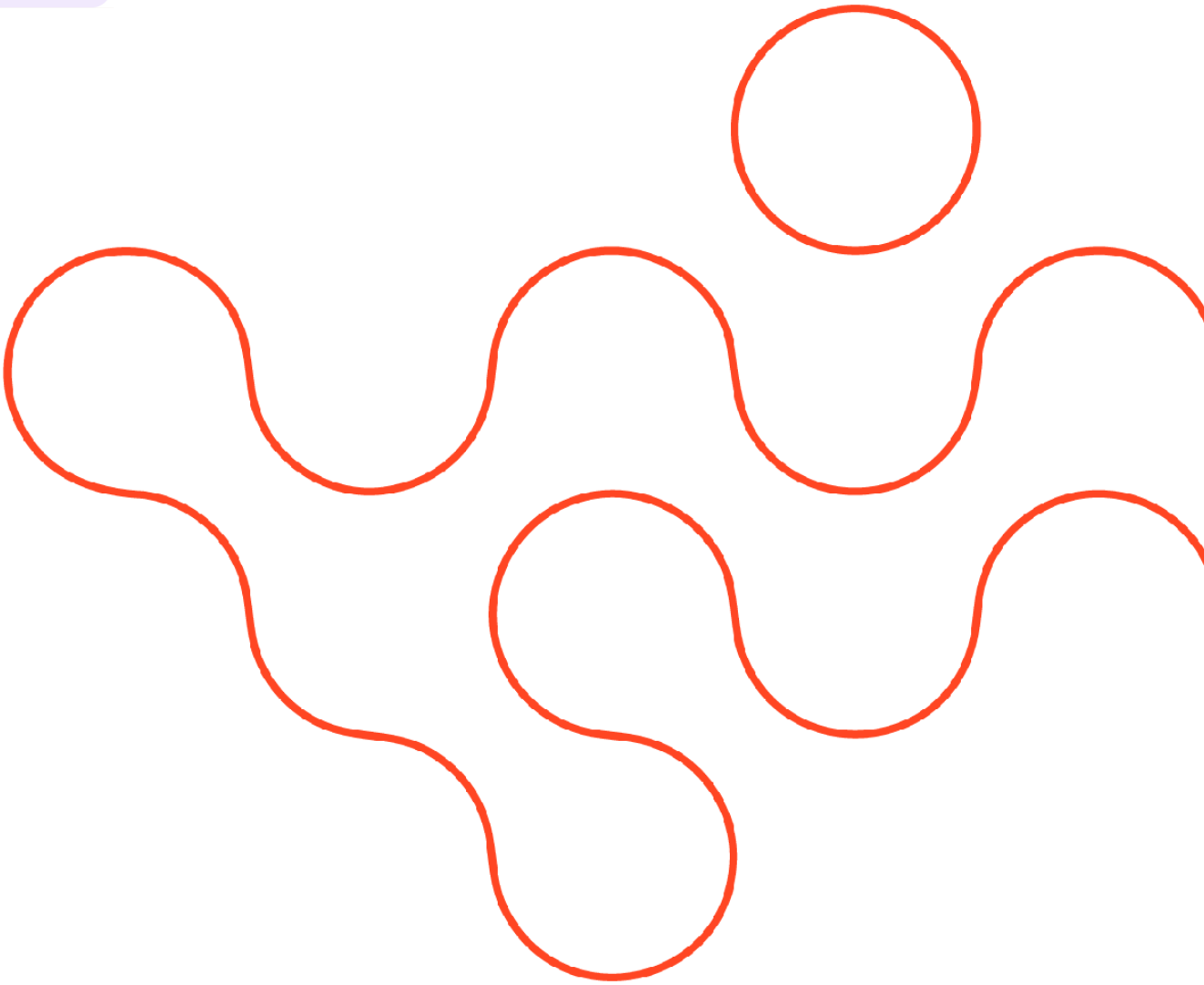
Product User Guide

Land Surface Temperature

Document Version 0.5.1 - September 4, 2026

support@ororatech.com | ororatech.com

Beta



Contents

1. Introduction

This user guide describes OroraTech's Land Surface Temperature (LST) product, detailing the product specifications, characteristics and performance metrics.

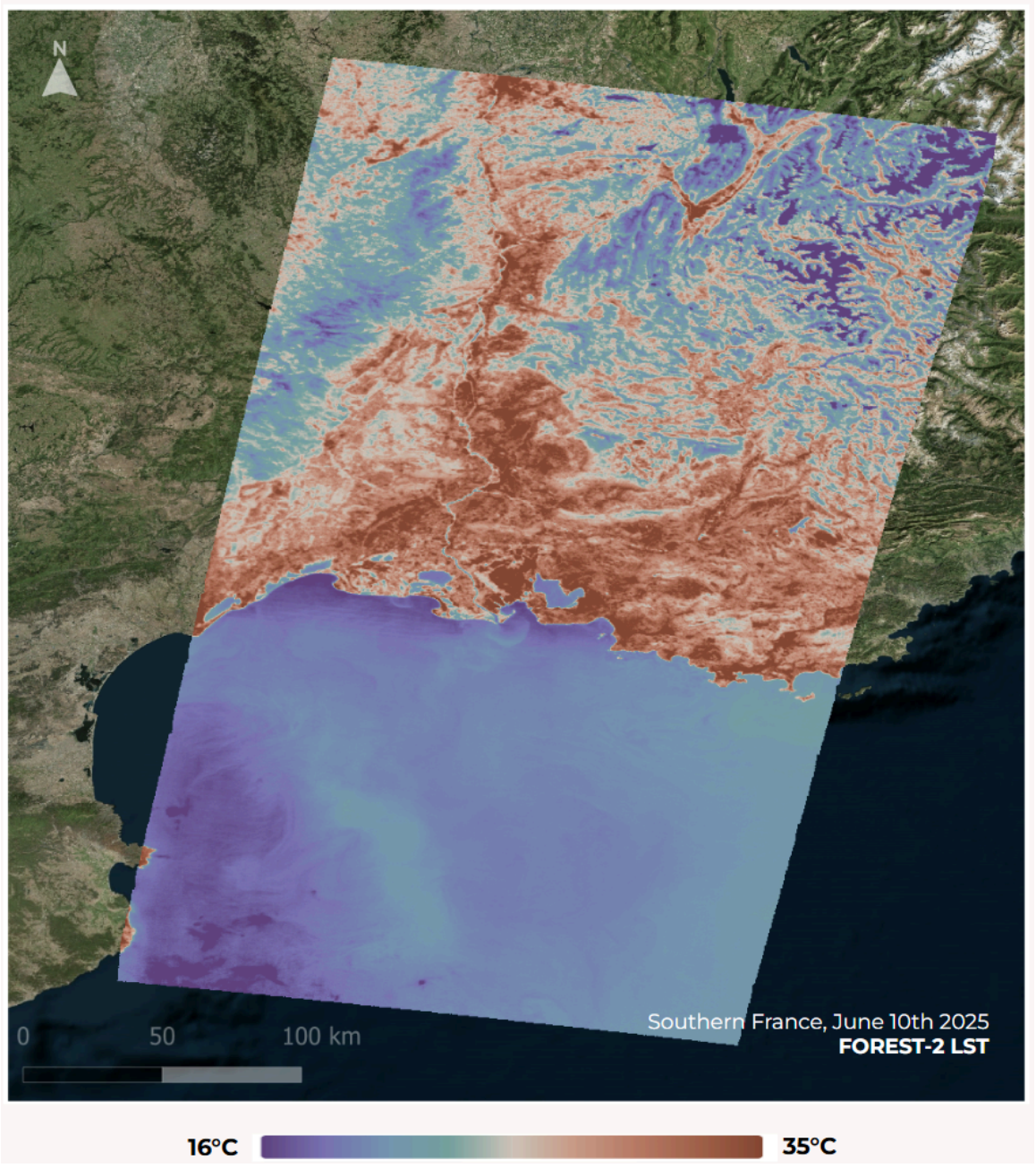


Figure 1: *OroraTech FOREST-2 LST over Southern France.*

1.1 Product Maturity - Closed Beta Phase

Beta

The OT-LST product is currently in a closed beta phase. Performance may vary, and the product may not be available at all times.

During this stage, we are continuously optimizing performance, accuracy, and coverage based on real-world data and user feedback.

However, the system is already operating at a level that delivers meaningful value to users, and we are actively working to further enhance its robustness and reliability ahead of full release.

All performance indicators throughout this description are described as targets. The product is delivered as-is.

2. Product Description

2.1. Specifications and Performance

This section describes the product performance and specifications of the LST product, which is summarized in Table 1.

Table 1: *Level-2 LST target product performance and specifications.*

Metric	Product Specifications
GSD	200 m
Swath (design)	210 km
Format	NetCDF (unprojected) GeoTiff (EPSG:4326)
Geolocation Accuracy (CE90)	<400 m <650 m (open ocean)
Temperature Accuracy	3K
Latency	NRT <24 h / NTC 7 days
Revisit	12 h (on average at equator)
Additional datasets	Image footprint (GeoJSON) Cloud mask

2.2. Data Quality Assessment

LST is independently assessed periodically by the European Space Agency's Optical Mission Performance Cluster (OPT-MPC) under the Earth Data Assessment Programme (EDAP) framework. The latest performance report can be provided on request. The validated satellites are listed in Table 2.

Table 2: *List of validated satellites and date of validation.*

Satellite	Sensor	Date of Validation
FOREST-2	IR1 & IR2	June 2025
FOREST-5	IR1 & IR2	March 2026

2.3. File Formats

To meet the needs of different user groups, the LST product is offered in two formats:

1. **Analytic Format (netCDF):**

Tailored for scientific and analytical applications, the netCDF files provide comprehensive data access. They include multiple data arrays such as surface temperature, geolocation, and data quality flags. A detailed description of these arrays is provided in Section 2.4. Metadata relevant to the product is also embedded in the main group, as outlined in Section 2.5. This format is unprojected.

2. **Visual Format (GeoTIFF):**

Optimized for visualization and mapping, the GeoTIFF files contain surface temperature data in Kelvin. The image footprint as well as key metadata are embedded in the attached GeoJSON file. This format is projected and referenced to EPSG:4326.

2.4. Product Data Arrays

The main data arrays are surface temperature values and the corresponding per-pixel coordinates given in the latitude (lat) and longitude (lon) arrays. Additional information about the observation geometry is given in the remaining arrays.

Table 3: *Level-2 LST product data arrays (netCDF).*

Variable	Dimensions	Data type	Units	Description
surface_temperature	[rows, columns]	float64	Kelvin	Land surface temperature
lat	[rows, columns]	float32	degrees north	Latitude per ground sample
lon	[rows, columns]	float32	degrees east	Longitude per ground sample
height	[rows, columns]	float32	meter	Height above WGS84 ellipsoid
flags	[rows, columns]	uint16	-	Quality flags per ground sample and band
observation_time_offset_first	[rows, columns]	float32	seconds	First observation time of the ground location relative to acquisition_time_start

Variable	Dimensions	Data type	Units	Description
observation_time_offset_last	[rows, columns]	float32	seconds	Last observation time of the ground location relative to acquisition_time_start
sensor_range	[rows, columns]	float32	meter	Distance from ground sample to sensor
sensor_azimuth_angle	[rows, columns]	float32	degrees	Azimuth angle of the ground-sample-to-sensor vector
sensor_zenith_angle	[rows, columns]	float32	degrees	Zenith angle of the ground-sample-to-sensor vector
solar_azimuth_angle	[rows, columns]	float32	degrees	Azimuth angle of the ground-sample-to-sun vector
solar_zenith_angle	[rows, columns]	float32	degree	Zenith angle of the ground-sample-to-sun vector
columns	scalar	int64	-	Number of columns
rows	scalar	int64	-	Number of rows

2.4.1. Data Quality Flags

The flags array contains data quality flags which are further specified in Table 4. The data flags represent a bitmask with each bit representing a different flag. If no flag is set for a given pixel, the data is considered to be valid. Note, that more than one flag can be set for a pixel. These flags are inherited from the Level-1C product. An example is given in Fig. 2.

Table 4: *Level-2 LST quality flags (netCDF).*

Bit	Flag	Description
0b0000001	OUTSIDE_BAND	Pixel was observed by at least one band, but not by all of them..
0b0000010	DEAD_PIXEL	Detector pixel is operating outside of specifications.
0b0000100	OUTSIDE_FRAME	Pixel does not contain observation data.
0b0001000	SATURATED	Pixel value is saturated.

Bit	Flag	Description
0b0010000	CLOUD	Pixel is covered by a cloud.
0b0100000	FIRE	Fire detection (only used in on-orbit fire detection)
0b1000000	RADIOMETRIC_FAILURE	Problem encountered in the radiometric processor returning a NaN value.
Remaining bits	reserved	Bits not used.

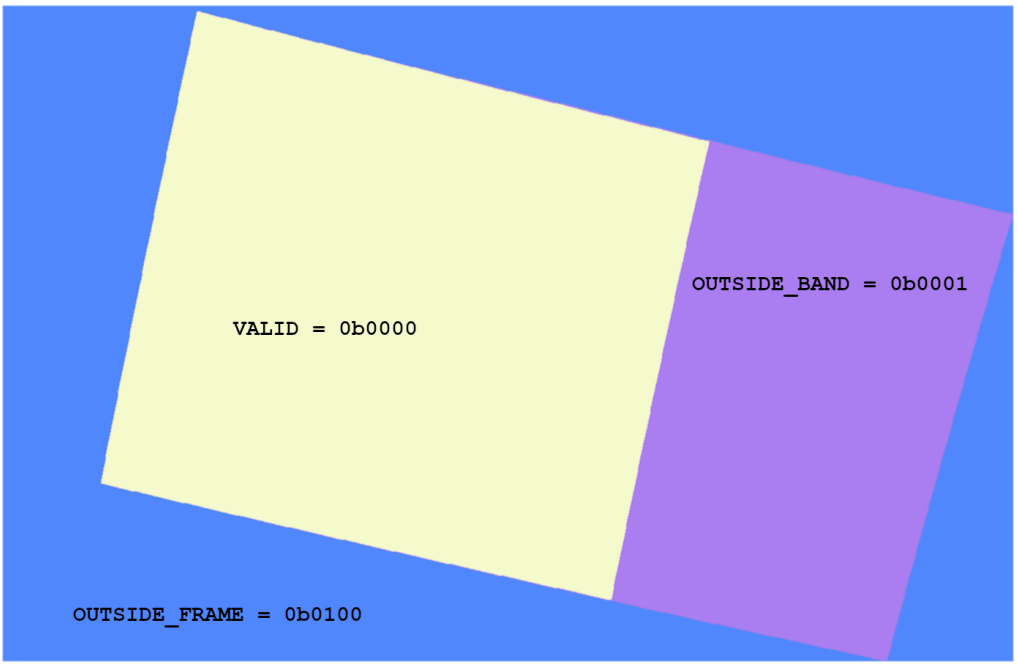


Figure 2: *Level-2 LST quality flags.*

2.4.2. Geometric Data Arrays

The definition of the geometric data arrays for sensor/solar azimuth and zenith angles is shown in Fig. 3. All values are calculated for the midpoint of the observation time range. The azimuth angle is in the range of 0° to 360° and is measured clockwise between north and the vector towards the satellite, i.e.

- north = 0°,
- east = 90°,
- south = 180° and
- west = -90°.

The zenith angle is in the range of 0° to 90° and is the angle between local zenith and the vector towards the satellite, i.e.

1. satellite at zenith above sample = 0°,
2. satellite at horizon = 90°.

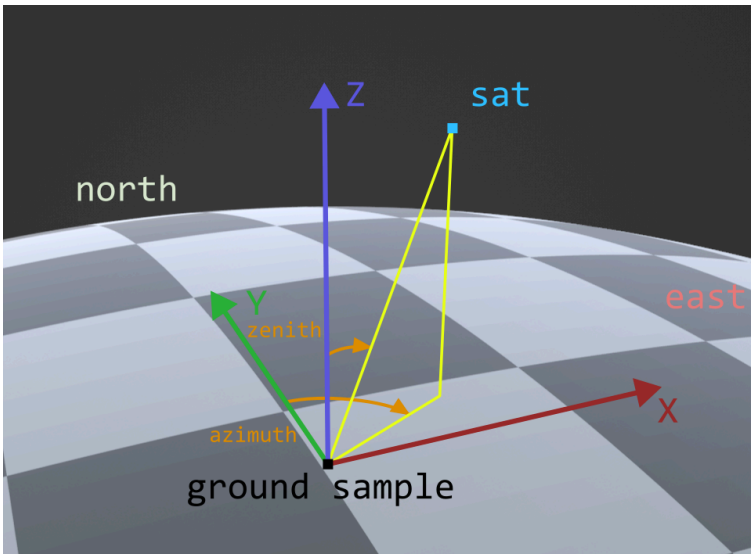


Figure 3: *Geometric parameters definition.*

2.5. Product Metadata

Next to the data arrays specified in Sec. 2.4, the LST product contains metadata attributes further described in Table 5.

Table 5: *Level-2 LST product metadata (netCDF).*

Attribute	Example value/format	Unit	Description
title	"OroraTech LST Data Product"	-	Title of the product
provider	"OroraTech"	-	Data provider
product_access_url	"https://ororatech.com/contact"		URL for OT products
platform	“F002”	-	Name of the satellite
mission	“F002”	-	Name of the mission
sensor	"F002-IR1"	-	Identifier of the sensor
sensor_type	“IR”	-	Spectral range of the imager (IR or VIS)
instrument	“SAFIRE”	-	Name of the payload
sensor_description	"Dual thermal infrared telescope with microbolometer detectors working in push-frame configuration"	-	Sensor description

Attribute	Example value/format	Unit	Description
acquisition_time_start	ISO 8601 format	-	Start time of the acquisition
acquisition_time_end	ISO 8601 format	-	End time of the acquisition
product_level	“Level-2”	-	Data processing level
reference_system	“EPSG:4326”	-	Geographic coordinate system of the grid
pixel_grid	"ORBIT_ALIGNED_GRID"	-	Type of grid used for storing the pixel information
algorithm_version	“0.2.0”	-	LST processor version
emissivity_version	“0.3.0”	-	Emissivity processor version
atmosphere_version	“0.3.0”	-	Atmosphere processor version
l1_version	“1.1.0”	-	Level 1 processor version
ground_sampling_distance	[200]	meter	Value of the ground sampling distance
platform_orbit_direction	“ASCENDING” / “DESCENDING”	-	Direction of the satellite movement in orbit during acquisition
auxiliary_data	['Copernicus Climate Change Service (C3S) (2017): ERA5: Fifth generation of ECMWF atmospheric reanalyses of the global climate . Copernicus Climate Change Service Climate Data Store (CDS), 2026-09-03. https://cds.climate.copernicus.eu/cdsapp#!/home', 'ESA WorldCover 10 m 2021 v200. https://doi.org/10.5281/zenodo.7254221', 'Copernicus Sentinel-2 (processed by ESA), 2021, MSI Level-2A BOA Reflectance Product.	-	Ancillary data used for processing the product

Attribute	Example value/format	Unit	Description
	Collection 1. European Space Agency. https://doi.org/10.5270/S2_-znk9xsj ', 'Copernicus Digital Elevation Model (DEM) was accessed on 2026-09-03 from https://registry.opendata.aws/copernicus-dem', 'Stewart ID, Oke TR. (2012). Local Climate Zones for Urban Temperature Studies. Bull Am Meteorol Soc. 93(12):1879-1900. https://doi.org/10.1175/BAMS-D-11-00019.1']		
band	“L2”	-	Names of the spectral bands used for generating the LST product
unit	“K”	-	Unit of the LST product
wavelength	[1.15e-05]	meter	Central wavelength of the spectral band
bandwidth	[1.804e-06]	meter	Bandwidth (FWHM) of the spectral band
footprint	<GeoJSON>	deg	Extent of the product specified as multipolygon with lon/lat coordinates
geolocation_accuracy_CE90	[198.0]	meter	90th percentile of the geolocation accuracy; *nan* if unknown, e.g. due to high cloud coverage
cloud_cover	<value between 0 and 1>	-	Relative cloud coverage

2.6. Product Naming Convention

The name of each product is composed of the following elements:

<platform>_<processing_level>_<mission_imager_id>_<product>_<acquisition_time>_<processed_time>_<recording_id>_<scene_id>.<extension>

Example:

F005_L2_IR1_LST_2025-07-04T024729_2026-03-20T162352_b1b73eb1-291f-42a9-aefe-f34d569b7f61_5c47a5.nc

The elements of the file name are specified in Table 6.

Table 6: *LST file name elements.*

platform	"F002"
processing level	"L2"
mission imager ID	"IR1" or "IR2"
product	"LST"
acquisition time	Sensing time of the sequence in ISO 8601 format
processed time	Processing time of the product in ISO 8601 format
recording ID	Recording identifier
scene ID	Scene of the recording identifier
extension	nc / tif

2.7. Product Generation

This section gives an overview of the processing steps involved in generating an LST product.

Inputs from Level-1C data

- Level-1C TOA radiance (L2 band)
- Satellite zenith angle
- Time of the acquisition

Atmospheric correction parameters

To model the self-emission and transmission of the atmosphere, ERA5 data from ECMWF is collected for the time and location of the acquisition. The software package libRadTran is then utilized to calculate the upwelling and downwelling atmospheric radiances, as well as the

transmittance of the atmosphere for the wavelength of the L2 band and the satellite zenith angle.

Emissivity calculation

The calculation of the emissivity for a given location is based on a combination of land cover classes and NDVI measurements as proposed by Valor and Caselles (1996). In a first step the emissivity is estimated using ESA WorldCover data. Then the NDVI is calculated from Sentinel-2 BOA reflectances and used to refine the emissivity.

Temperature retrieval

The land surface temperature is finally retrieved by inverting the radiative transfer equation and applying the atmospheric parameters and the emissivity.

2.8. Product Ordering

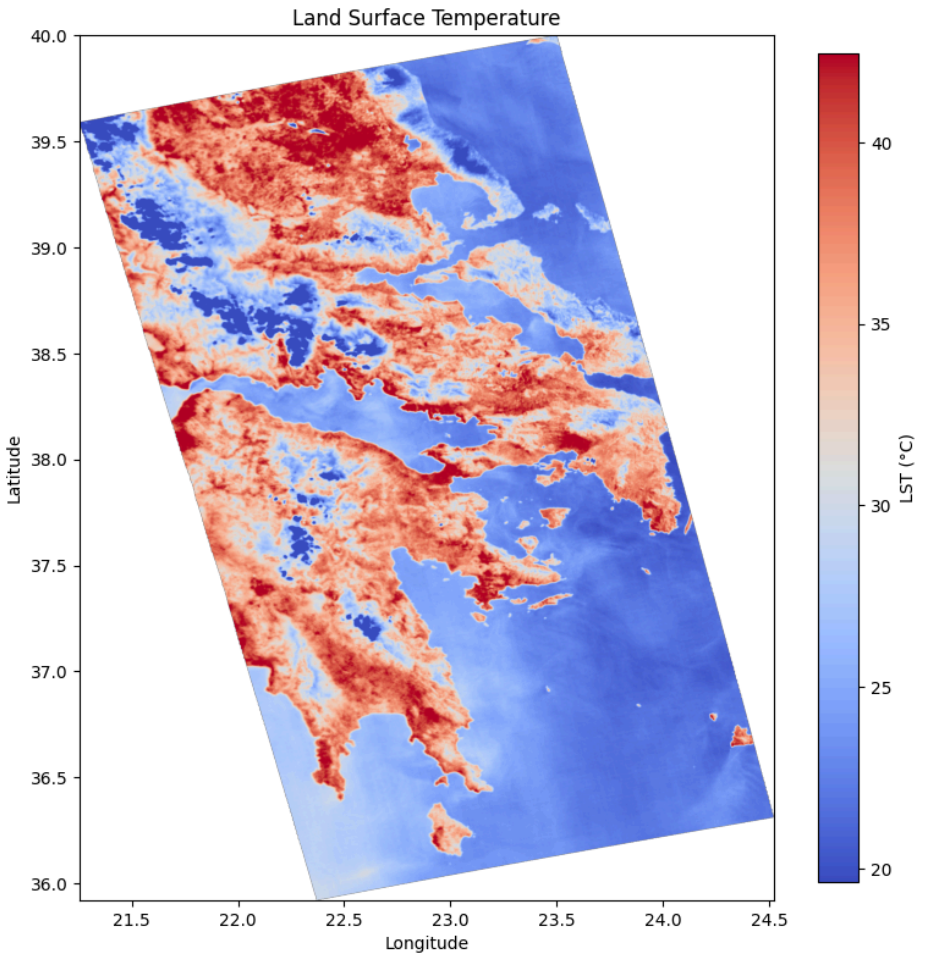
To order data products, please use the contact form on the OroraTech website:
https://ororatech.com/contact

3. Python Examples

3.1 Loading the LST GeoTIFF and plotting it

```
import numpy as np
import rasterio
import matplotlib.pyplot as plt

geotiff_path =
"/F005_L2_IR1_LST_2026-07-28T135715_2026-07-29T151129_6a53b336-7eb1-44e8-961a-f985ddf5
e6f7_68353f.tif"

# --- Open the file and read the LST band ---
with rasterio.open(geotiff_path) as src:
    lst_kelvin = src.read(1)          # 2D array, shape (rows, cols)
    transform = src.transform         # affine georeferencing transform
    crs = src.crs                     # coordinate reference system
    bounds = src.bounds               # (left, bottom, right, top)

print(f"CRS: {crs}")
print(f"Shape: {lst_kelvin.shape}, dtype: {lst_kelvin.dtype}")
print(f"Bounds (lon/lat): {bounds}")

# --- Convert to Celsius for readability (optional) ---
lst_celsius = lst_kelvin - 273.15

# --- Plot ---
fig, ax = plt.subplots(figsize=(8, 10))
img = ax.imshow(
    lst_celsius,
    cmap="coolwarm",
    extent=(bounds.left, bounds.right, bounds.bottom, bounds.top),
    vmin=np.nanpercentile(lst_celsius, 2),
    vmax=np.nanpercentile(lst_celsius, 98),
)
ax.set_title("Land Surface Temperature")
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
cbar = fig.colorbar(img, ax=ax, shrink=0.7)
cbar.set_label("LST (°C)")
plt.tight_layout()
plt.savefig("lst_geotiff.png", dpi=150)
plt.show()
```

Output:

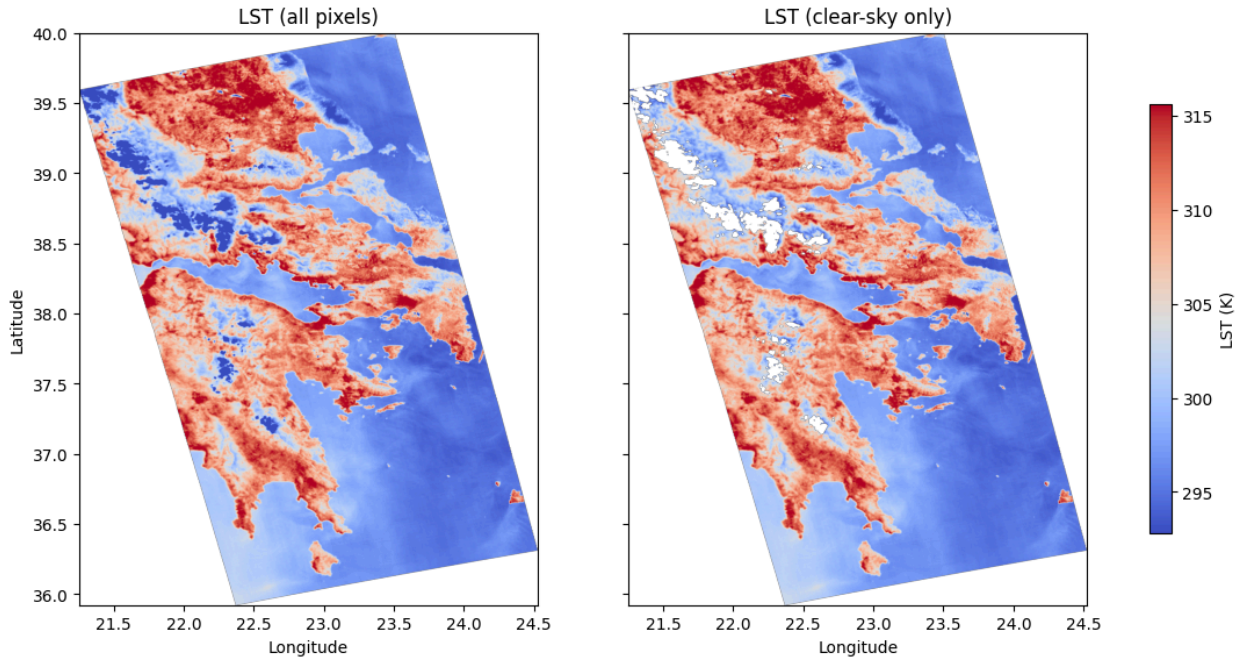


3.2 Cloud Masking the LST GeoTIFF

```
import numpy as np
import rasterio
import matplotlib.pyplot as plt

lst_path                                                                    =
"/F005_L2_IR1_LST_2026-07-28T135715_2026-07-29T151129_6a53b336-7eb1-44e8-961a-f985ddf5
e6f7_68353f.tif"
cloudmask_path                                                              =
"/F005_L2_IR1_LST_2026-07-28T135715_2026-07-29T151129_6a53b336-7eb1-44e8-961a-f985ddf5
e6f7_68353f_cloudmask.tif"

# --- Open both rasters ---
with rasterio.open(lst_path) as src:
    lst_kelvin = src.read(1)
    lst_profile = src.profile
    bounds = src.bounds

with rasterio.open(cloudmask_path) as src:
    cloud_mask = src.read(1)   # 1 = cloud, 0 = clear
```

```
# --- Mask out cloudy pixels ---
# Keep only clear-sky pixels; set cloudy pixels to NaN
lst_clear = np.where(cloud_mask == 1, np.nan, lst_kelvin)

n_total = lst_kelvin.size
n_cloud = int((cloud_mask == 1).sum())
n_clear_valid = int(np.isfinite(lst_clear).sum())
print(f"Total pixels:        {n_total}")
print(f"Cloudy pixels:       {n_cloud} ({100 * n_cloud / n_total:.1f}%)")
print(f"Clear & valid LST:   {n_clear_valid} ({100 * n_clear_valid / n_total:.1f}%)")

# --- Save the clear-sky LST as a new GeoTIFF ---
clear_profile = lst_profile.copy()
clear_profile.update(nodata=np.nan)

with rasterio.open("lst_clearsky.tif", "w", **clear_profile) as dst:
    dst.write(lst_clear.astype(np.float32), 1)

# --- Plot before/after side by side ---
fig, axes = plt.subplots(1, 2, figsize=(14, 8), sharey=True)
extent = (bounds.left, bounds.right, bounds.bottom, bounds.top)
vmin, vmax = np.nanpercentile(lst_kelvin, [2, 98])

axes[0].imshow(lst_kelvin, cmap="coolwarm", extent=extent, vmin=vmin, vmax=vmax)
axes[0].set_title("LST (all pixels)")

im = axes[1].imshow(lst_clear, cmap="coolwarm", extent=extent, vmin=vmin, vmax=vmax)
axes[1].set_title("LST (clear-sky only)")

for ax in axes:
    ax.set_xlabel("Longitude")
axes[0].set_ylabel("Latitude")

fig.colorbar(im, ax=axes, shrink=0.6, label="LST (K)")
plt.savefig("cloudmask_clip.png", dpi=150, bbox_inches="tight")
plt.show()
```

Output:



3.3 Exploring the LST NetCDF: Layers, Metadata, and Plots

```
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

nc_path                                                                                                =
"/F005_L2_IR1_LST_2026-07-28T135715_2026-07-29T151129_6a53b336-7eb1-44e8-961a-f985ddf5
e6f7_68353f.nc"

# --- Open the dataset ---
ds = xr.open_dataset(nc_path)

# --- 1) List the data variables (layers) ---
print("Data variables in this file:")
for name, var in ds.data_vars.items():
    units = var.attrs.get("units", "-")
    long_name = var.attrs.get("long_name", "-")
       print(f"    -  {name:<30}  shape={var.shape}    dtype={var.dtype}    units={units}
({long_name})")

# --- 2) Print the global attributes as a table ---
print("\nGlobal attributes:")
key_width = max(len(k) for k in ds.attrs) + 2
print(f"{'Attribute':<{key_width}} Value")
print("-" * (key_width + 60))
for key, value in ds.attrs.items():
    # Long values (lists, dicts) are trimmed for table display
    value_str = str(value)
    if len(value_str) > 80:
        value_str = value_str[:77] + "..."
```

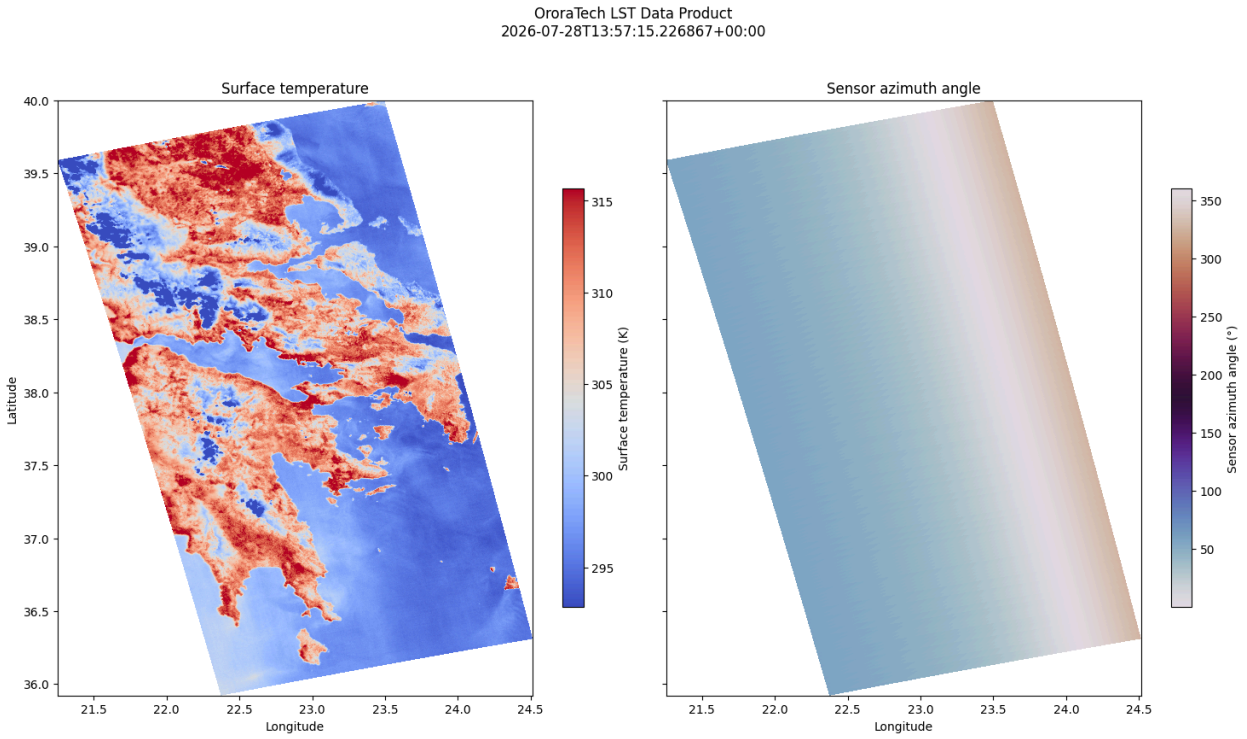
```
    print(f"{key:<{key_width}} {value_str}")

# --- 3) Plot surface temperature and sensor azimuth angle side by side ---
valid_cols = ~np.isnan(ds.lat.values).any(axis=1)
ds_plot = ds.isel(columns=valid_cols)

lat = ds_plot.lat.values
lon = ds_plot.lon.values
lst_kelvin = ds_plot.surface_temperature.values
sensor_azimuth = ds_plot.sensor_azimuth_angle.values

fig, axes = plt.subplots(1, 2, figsize=(15, 9), sharey=True)

mesh0 = axes[0].pcolormesh(
    lon, lat, lst_kelvin,
    cmap="coolwarm",
    shading="auto",
    vmin=np.nanpercentile(lst_kelvin, 2),
    vmax=np.nanpercentile(lst_kelvin, 98),
)
axes[0].set_title("Surface temperature")
axes[0].set_xlabel("Longitude")
axes[0].set_ylabel("Latitude")
axes[0].set_aspect("equal")
fig.colorbar(mesh0, ax=axes[0], shrink=0.6, label="Surface temperature (K)")

mesh1 = axes[1].pcolormesh(
    lon, lat, sensor_azimuth,
    cmap="twilight",
    shading="auto",
    vmin=np.nanmin(sensor_azimuth),
    vmax=np.nanmax(sensor_azimuth),
)
axes[1].set_title("Sensor azimuth angle")
axes[1].set_xlabel("Longitude")
axes[1].set_aspect("equal")
fig.colorbar(mesh1, ax=axes[1], shrink=0.6, label="Sensor azimuth angle (°)")

fig.suptitle(f"{ds.attrs['title']}\n{ds.attrs['acquisition_time_start']}")
plt.tight_layout()
plt.savefig("netcdf_variables_attrs_plot.png", dpi=150, bbox_inches="tight")
plt.show()

ds.close()
```

Output:

```
Data variables in this file:

  - lat                            shape=(1028, 2096)  dtype=float32  units=degrees_north
(latitude)
```

```
  - lon                              shape=(1028, 2096)  dtype=float32  units=degrees_east
(longitude)

  - surface_temperature              shape=(1028, 2096)  dtype=float32  units=K  (land surface
temperature)

  - height                           shape=(1028, 2096)  dtype=float32  units=meters  (height
above WGS84)

  - flags                            shape=(1028, 2096)  dtype=float32  units=-  (-)

  - observation_time_offset_first    shape=(1028, 2096)  dtype=float32  units=seconds (unix
timestamp)  (-)

  - observation_time_offset_last     shape=(1028, 2096)  dtype=float32  units=seconds (unix
timestamp)  (-)

  - sensor_range                     shape=(1028, 2096)  dtype=float32  units=meters  (distance
from target location to sensor)

  - sensor_azimuth_angle             shape=(1028, 2096)  dtype=float32  units=degrees  (Sensor
azimuth angle at target location)

  - sensor_zenith_angle              shape=(1028, 2096)  dtype=float32  units=degrees  (Sensor
zenith angle at target location)

  - solar_azimuth_angle              shape=(1028, 2096)  dtype=float32  units=degrees  (Solar
azimuth angle at target location)

  - solar_zenith_angle               shape=(1028, 2096)  dtype=float32  units=degrees  (Solar
zenith angle at target location)


Global attributes:

Attribute                   Value

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

title                       OroraTech LST Data Product

provider                    OroraTech

product_access_url          https://ororatech.com/contact

platform                    F005

mission                     OTC-P1

sensor                      F005-IR2

sensor_type                 IR

instrument                  SAFIRE

sensor_description          Dual thermal infrared telescope with microbolometer detectors
working in push...

acquisition_time_start      2026-07-28T13:57:15.226867+00:00

acquisition_time_end        2026-07-28T13:58:36.885671+00:00

product_level               Level-2

reference_system            EPSG:4326

pixel_grid                  ORBIT_ALIGNED_GRID

algorithm_version           0.2.0-unknown
```

```
emissivity_version          0.3.0-unknown

atmosphere_version          0.3.3-unknown

l1_version                  1.1.0

ground_sampling_distance    200

platform_orbit_direction    ASCENDING

auxiliary_data              ['Copernicus Climate Change Service (C3S) (2017): ERA5: Fifth
gen...

band                        L2

unit                        K

wavelength                  1.153e-05

bandwidth                   2.045e-06

footprint                   {'type': 'MultiPolygon', 'coordinates': [[[[23.483, 36.123],
[22.369, 35.921]...

geolocation_accuracy_CE90   340.0

cloud_cover                 0.0274
```



End of document