SIFT – An interactive tool for satellite data visualization and analysis

Andrea Meraner, Johan Strandgren,
Sauli Joro, Guillaume Aubert
EUMETSAT, Darmstadt, Germany

David Hoese[1], Ray Garcia[1], Scott Lindstrom[1], Jordan Gerth[2]
1Space Science and Engineering Center, University of Wisconsin, Madison, USA
2National Weather Service Office of Observations, Silver spring, USA

Alexander Rettig, Max Kreischer, Alexandra Melzer,
Nicolai Kellerer, Paul Pazderski, Florian Schröder
ask – Innovative Visualisierungslösungen GmbH, Darmstadt, Germany

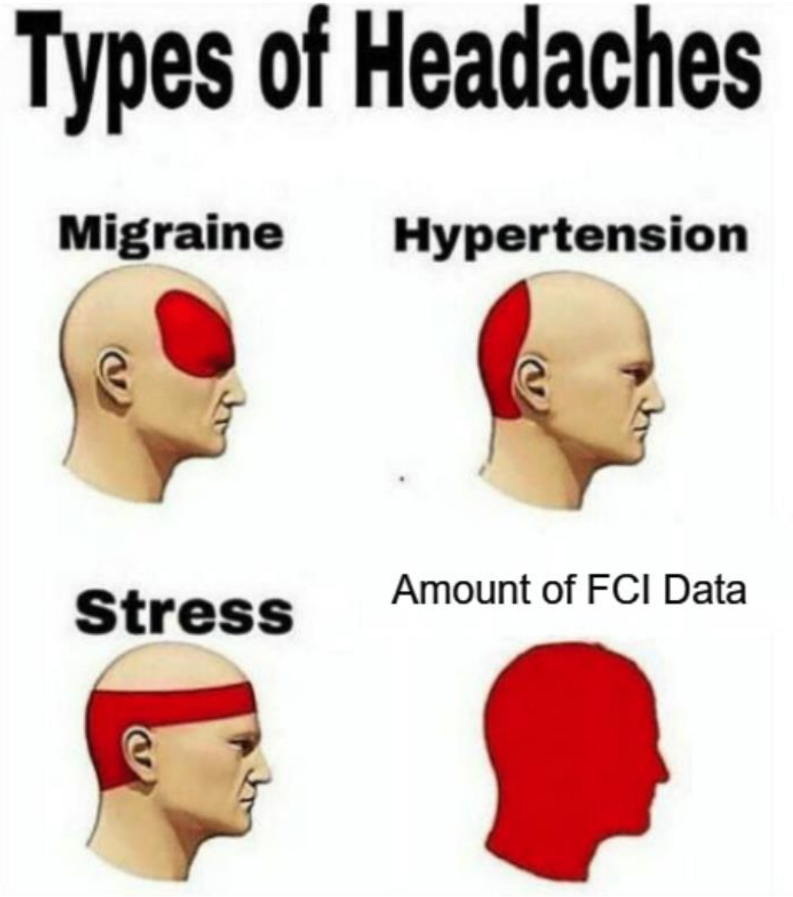
LI Short Course 2024

SIFT: Background and Challenges

- Modern imagers produce an enormous amount of data to be handled and visualized
  - Clear need of new strategy and technologies for data visualization

- Due to the heterogeneous types of data used for e.g. MTG-I Cal/Val activities,
  a multi-mission, modular and flexible approach was desirable

- Combination of two existing open source software solutions showed the most promising solution:
  - SIFT as graphic and visualization engine
  - Pytroll as library for data reading and processing



SIFT: Visualization Engine

https://sift.ssec.wisc.edu/
https://github.com/ssec/sift

- *Satellite Information Familiarization Tool*, initially developed at SSEC University of Wisconsin for the use by their trainers:
  - Cross OS (Linux, Win, MacOS)
  - Designed to be fast and to cope with high-resolution imager datasets (data thinning and GPU acceleration)

- EUMETSAT Vision: SIFT shall be an easy to use and responsive multi-mission data analyses and visualization application supporting many different use cases:
  - *Cal/Val*
  - *science*
  - *training*
  - *satellite operations, ...*

- To achieve this, EUMETSAT, together with *ask**, developed a new version of SIFT, initially focusing on MTG-I needs

https://askvisual.de/

** ask – Innovative Visualisierungslösungen GmbH*

SIFT data access and processing engine: Pytroll

- Pytroll is a python framework for the reading and processing of Earth observation satellite data. It implements the most common operations needed for satellite data handling:
  - Product readers
  - Reprojection, resampling, overlay of cartographic features
  - Generation of RGBs, geometric/atmospheric corrections, …

- SIFT v2.0 takes advantage of the reading and resampling capabilities of the Pytroll packages Satpy and Pyresample to import data into the visualization engine:
  - A new Satpy reader can directly be utilized by SIFT
  - All pyresample-resamplers are available
  - All satpy composites are directly available

PyTROLL

http://pytroll.github.io/

https://github.com/pytroll/

Some (!) available Satpy readers

EUMETSAT data

avhrr_l1b
fci_l1c_nc
fci_l2_nc
iasi_l2
li_l2
mviri_l1b_fiduceo
olci_l1b
olci_l2
seviri_l1b
seviri_l2_grib
seviri_l2_bufr
slstr_l1b
slstr_l2
vii_l1b_nc
vii_l2_nc

Other data

abi_l1b
abi_l2_nc
ahi_hrit
ahi_hds
amsr2_l1b
amsr2_l2
atdnet
caliop_l2_cloud
cmsaf_claas2_l2_nc
euclid
gld360
glm_l2
goes-imager_hrit
goes-imager_nc
modis_l1b
modis_l2
msi_safe
nwcsaf-geo
nwcsaf-pps_nc
tropomi_l2
viirs_l1b

First release of SIFT v2.0

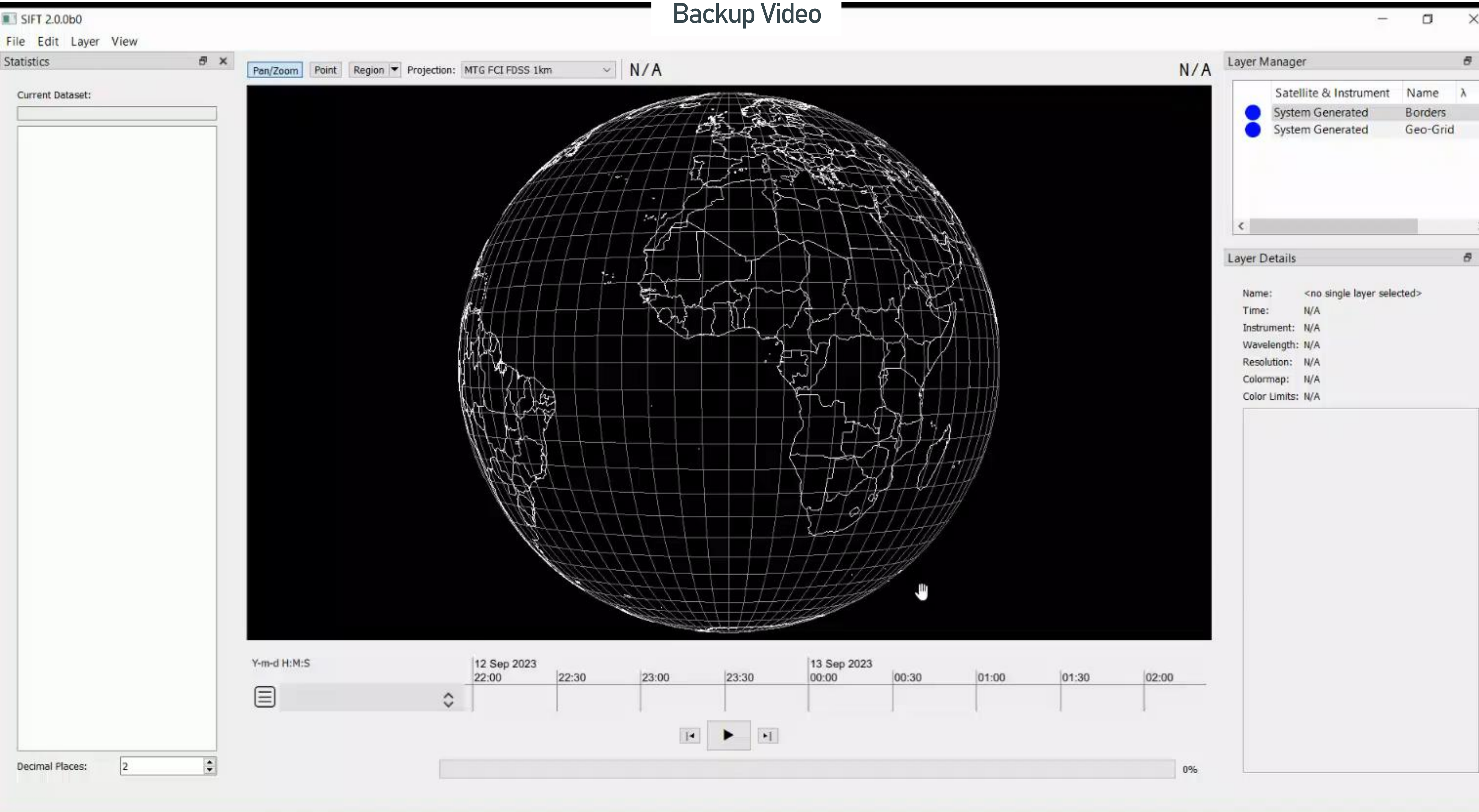
- SIFT v2.0 contains all EUMETSAT-led developments, targeting MTG-I commissioning and MTG-UP

- Main new features are:
  - Full support for GEO, LEO and point data, through Satpy integration
  - support for composite (RGB) visualization
  - an improved timeline manager
  - integration of a statistics module
  - full resampling functionalities using Pyresample
  - an automatic update/monitoring mode
  - partial redesign of the UI/UX
  - … many more small but useful features!

SIFT Live Demo

🤞

Backup Video



Current Dataset:

Projection: MTG FCI FDSS 1km

	Satellite & Instrument	Name	λ
●	System Generated	Borders	
●	System Generated	Geo-Grid	

Name: <no single layer selected>
Time: N/A
Instrument: N/A
Wavelength: N/A
Resolution: N/A
Colormap: N/A
Color Limits: N/A

Decimal Places: 2

A note on computational requirements

- SIFT runs on Win, Mac and Linux, but depends on many complex libraries
- Can make use of GPU via PyOpenGL and Vispy
- Simpler to setup on a local system, virtual machines/remote servers are trickier due to limitations in displaying OpenGL

- Lower end specs to run basic SIFT functions (but, the more the merrier!)
  - Windows 10+ / Mac OS X >11.0 / Linux >= Rocky Linux 8
  - 8GB RAM
  - Disk space (preferably on a SSD/NVMe drive) with 20GB+ available
  - GPU with 2GB VRAM and OpenGL 3+ support
  - Data files to be loaded can require several GB of disk space

First release of SIFT v2.0

- Currently in **beta** release since May 2023
  - **Thank you to all people trying it out and reporting issues in their use cases/workflows!**
  - Some major issues have been identified and are still open
    - Colormap "discretises" when stretching on narrow ranges
    - MacOS still problematic
    - Transparency of datasets still problematic
    - ~~Image export functionality not working~~ (fixed in v2.0.0b1 released… yesterday ☺)
  - Note that SIFT is not an official EUM tool and is developed on a best effort basis.
  - Contributions are always welcome!

→ We are of course also still happy about any issue/bug reports!*
- Preferred way is through a Github issue: https://github.com/ssec/sift
- Or open a thread in the uwsift Google Group: https://groups.google.com/g/uwsift
- Or contact us on https://gitter.im/ssec/sift

Further Info

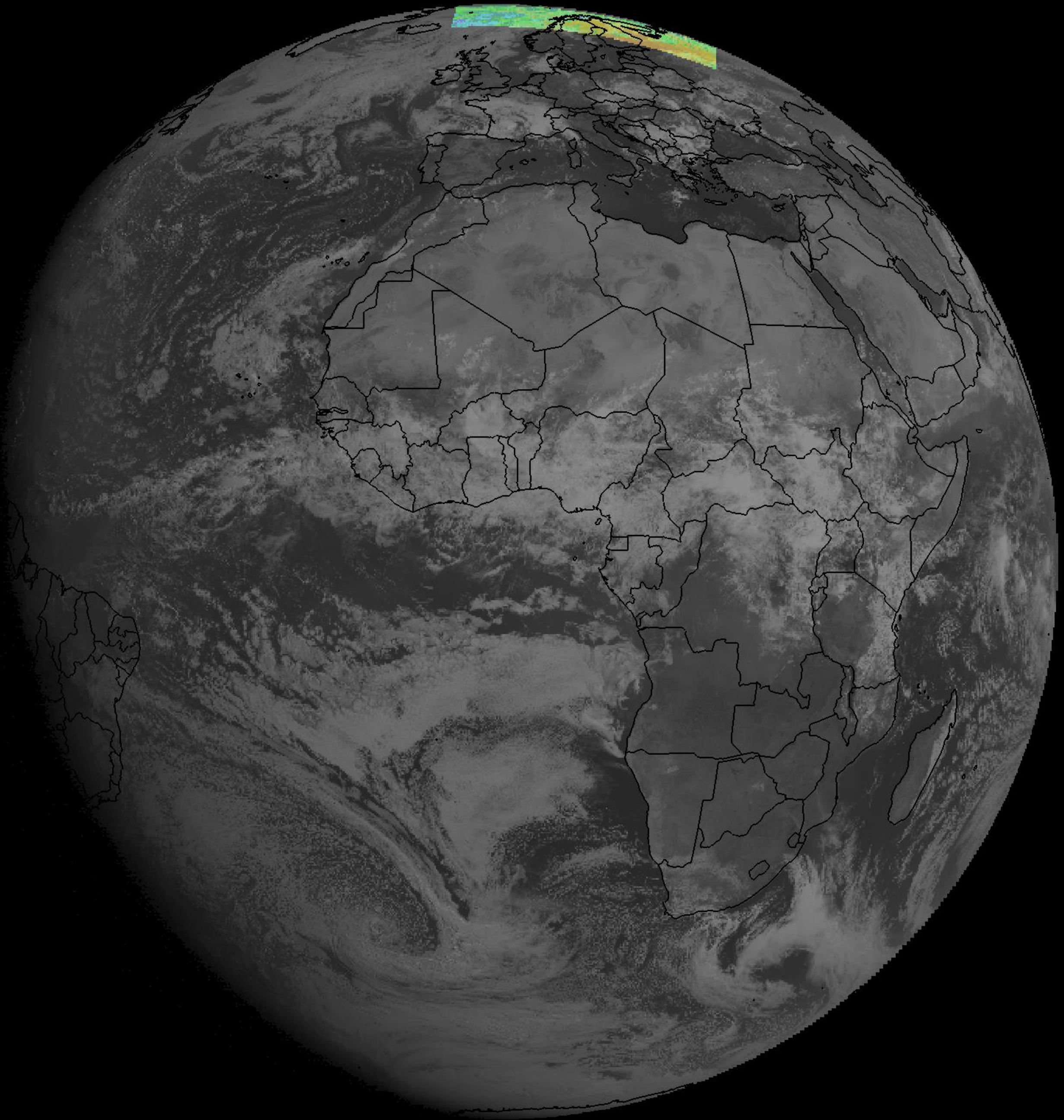
- Download link on ftp: https://bin.ssec.wisc.edu/pub/sift/dist/experimental/
  New builds are uploaded regularly with bugfixes and satpy updates

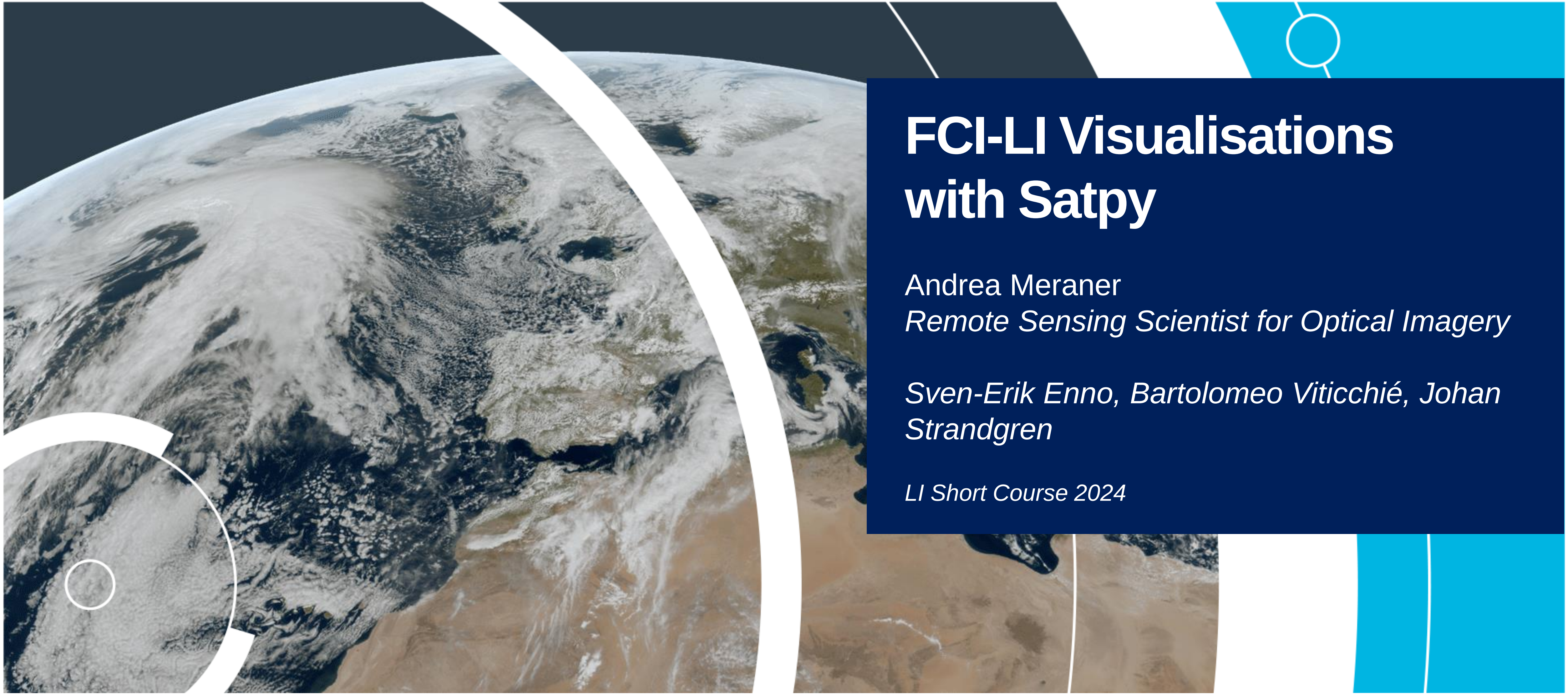
- Updated (configuration) documentation is on ReadTheDocs: https://sift.readthedocs.io/en/latest/
- Wiki, User Manual, and website will be gradually updated

- SIFT short course webpage: https://classroom.eumetsat.int/course/view.php?id=478
  (or just search for "sift eumetsat short course")
  - Contains recording of **extensive demo**, including LEO and L2
  - Contains links to test and demo data

- **SIFT Github page for more links and instructions:** https://github.com/ssec/sift

Thank you for your attention!





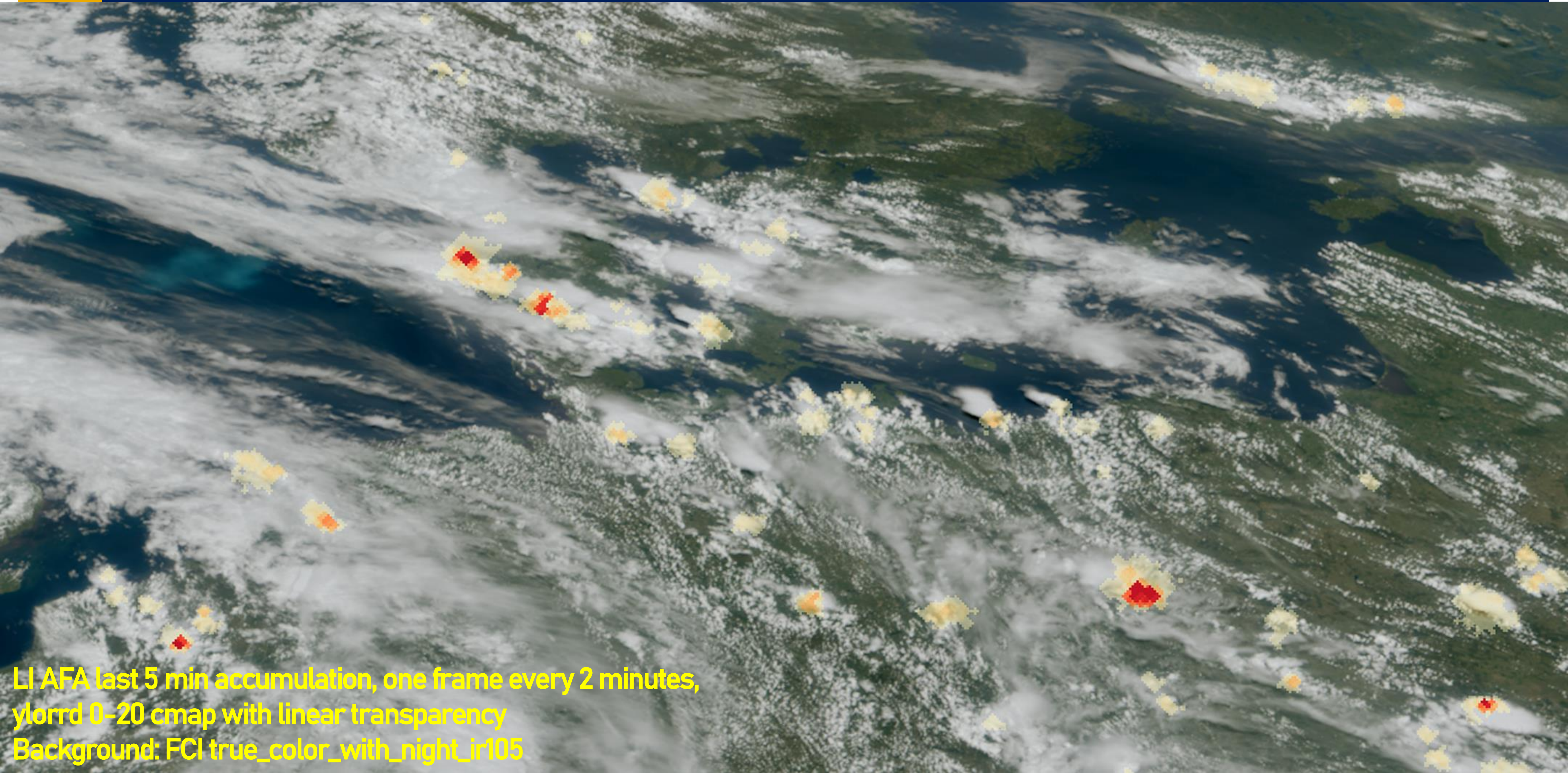
FCI-LI Visualisations with Satpy

Andrea Meraner
Remote Sensing Scientist for Optical Imagery

Sven-Erik Enno, Bartolomeo Viticchié, Johan Strandgren

LI Short Course 2024

Example: Merging LI with FCI



LI AFA last 5 min accumulation, one frame every 2 minutes, ylorrd 0-20 cmap with linear transparency
Background: FCI true_color_with_night_ir105

First Experiments with 2-d Accumulated Products

```
# FCI
composites:
  true_color_with_night_ir105_acc_flash_area:
    compositor:
!!python/name:satpy.composites.BackgroundCompositor
    standard_name: imager_with_lightning
    prerequisites:
      - acc_flash_area_alpha
      - true_color_with_night_ir105

enhancements:
  imager_with_lightning:
    standard_name: imager_with_lightning
    operations: []
```

```
# LI
composites:
   acc_flash_area_alpha:
    compositor:
!!python/name:satpy.composites.SingleBandCompositor
    standard_name: acc_flash_area_alpha
    prerequisites:
    - accumulated_flash_area

enhancements:
  acc_flash_area_alpha:
    standard_name: acc_flash_area_alpha
    operations:
    - name: colorize
      method: !!python/name:satpy.enhancements.colorize
      kwargs:
        palettes:
        - {colors: ylorrd, min_value: 0, max_value: 20,
           alpha_value_min: 100, alpha_value_max: 255}
```

First Experiments with 2-d Accumulated Products, x LI files

```
def nan_sum(datasets):
    attrs = combine_metadata(*[data_arr.attrs for data_arr in datasets])
    concat_ds = xr.concat(datasets, dim="sum_dim")
    sum_ds = concat_ds.sum(dim="sum_dim", skipna=True, min_count=1, keep_attrs=True)
    sum_ds.attrs = attrs
    return sum_ds


def plot_li_acc_with_background(li_filenames, fci_filenames, output_folder, dataset_names):

    li_ms = MultiScene.from_files(li_filenames, reader='li_l2_nc')
    li_ms.load(['flash_area'], upper_right_corner='NE')
    li_scn_b = li_ms.blend(nan_sum)


    fci_scene = Scene(filenames=fci_filenames, reader='fci_l1c_nc')
    fci_scene['flash_area'] = li_scn_b['flash_area']

    fci_scene.load(dataset_names, upper_right_corner='NE')
    ms_r = fci_scene.resample('mtg_fci_fdss_1km', resampler='native')
    ms_lcl = ms_r.crop(ll_bbox=[1.5, 49, 20.26, 62])
    ms_lcl.save_datasets(datasets=dataset_names,
                         filename=output_folder +
                                  f'/FCI-LI_{{name}}_'
                                  f'{fci_scene["flash_area"].attrs["start_time"].strftime("%Y%m%d-%H%M%S")}_'
                                  f'{fci_scene["flash_area"].attrs["end_time"].strftime("%Y%m%d-%H%M%S")}.png’,
                      )

# Plus some code implementing the rolling window logic
# using find_files_and_readers with start and end_time

# accumulating last 5 minutes every 2 minutes, showing last available FCI Scene
```

More Information:

- Full code to reproduce and first version of composites:
  https://github.com/pytroll/satpy/pull/2853

- Satpy Documentation and Code:
  https://github.com/pytroll/satpy
  https://satpy.readthedocs.io/en/latest/

- More Info on Pytroll Project (contributions are always welcome!)
  https://pytroll.github.io/

Thank you!

Questions are welcome.

andrea.meraner on Pytroll Slack
ameraner on GitHub
andrea.meraner@eumetsat.int