NMRF/TR/08/2020

TECHNICAL REPORT

Data Processing and Visualisation of NOAA and MetOp Satellite Data

Srinivas Desamsetti, C. J. Johny, M. Sateesh, M. N. R. Sreevathsa and V. S. Prasad

July 2020

National Centre for Medium Range Weather Forecasting
Ministry of Earth Sciences, Government of India
A-50, Sector-62, Noida-201 309, INDIA

Data Processing and Visualisation of NOAA and MetOp Satellite Data

Srinivas Desamsetti, C. J. Johny, M. Sateesh,
M. N. R. Sreevathsa and V. S. Prasad

July 2020

National Centre for Medium Range Weather Forecasting
Ministry of Earth Sciences
A-50, Sector 62, Noida-201 309, INDIA

Ministry of Earth Sciences
National Centre for Medium Range Weather Forecasting
Document Control Data Sheet

1	Name of the Institute	National Centre for Medium Range Weather Forecasting
2	Document Number	NMRF/TR/08/2020
3	Date of publication	July, 2020
4	Title of the document	Data Processing and Visualisation of NOAA and MetOp Satellite Data
5	Type of Document	Technical Report
6	No. of Pages, Figures and Tables	44 pages, 2 Tables, 12 Figures, 9 Annexures.
7	Number of References	7
8	Author (S)	Srinivas Desamsetti, C.J. Johny, M. Sateesh, M. N. R. Sreevathsa and V. S. Prasad
9	Originating Unit	NCMRWF
10	Abstract	NCMRWF receives level-0 data from the low earth orbit satellites in near real time from INCOIS and level-1b data from other Direct Broadcast Networks of WMO. The level-0 mainly consists of data from NOAA and MetOp series of satellites. An automated data processing chain (ADPC) to process this level-0 data using AAPP, and MetOpizer software tools has been established at NCMRWF. It has capability to generate level-1b, level-1c and level-1d data from AVHRR, AMSU-A, MHS, HIRS and IASI sensors which are onboard of above-mentioned satellites. The three sensors i.e., AMSU-A, MHS and HIRS together form the ATOVS. The ADPC provides data products for use in the data assimilation systems of NWP models at NCMRWF; and for generating geospatial imageries of AVHRR, AMSU-A and MHS for forecasters at IMD. This report describes the various software tools used to make this ADPC and subsequent geospatial visualization.
11	Security classification	Non-Secure
12	Distribution	Unrestricted Distribution
13	Keywords	Decoding, monitoring, NOAA, MetOp, AAPP, visualization

Table of Contents

1 Introduction 4

2 Data 4

3 Methodology 5

3.1 AAPP 5

3.2 MetOpizer 8

4 Data Processing 8

4.1 NOAA HRPT Data 9

4.2 MetOp AHRPT Data 9

5 Data Visualization 10

6 Data Monitoring 11

7 Acknowledgements 12

8 References 12

9 Figures 13

10 ANNEXURES 22

Annexure-I: submit_hrpt_process.sh 22

Annexure-II: noaa_hrpt_process.sh 23

Annexure-III: metop_hrpt_process.sh 26

Annexure-IV: run_l1b_plot.sh 31

Annexure-V: run_l1c_plot.sh 33

Annexure-VI: plot_hrpt_l1b_mas.py 36

Annexure–VII: aapp1c_quicklook.py 38

Annexure–VIII: run_plot_time_delays.sh 42

Annexure–IX: plt_delay.r 44

Abstract

NCMRWF receives level-0 data from the low earth orbit satellites in near real time from INCOIS and level-1b data from other Direct Broadcast Networks of WMO. The level-0 mainly consists of data from NOAA and MetOp series of satellites. An automated data processing chain (ADPC) to process this level-0 data using AAPP, and MetOpizer software tools has been established at NCMRWF. It has capability to generate level-1b, level-1c and level-1d data from AVHRR, AMSU-A, MHS, HIRS and IASI sensors which are onboard of above-mentioned satellites. The three sensors i.e., AMSU-A, MHS and HIRS together form the ATOVS. The ADPC provides data products for use in the data assimilation systems of NWP models at NCMRWF; and for generating geospatial imageries of AVHRR, AMSU-A and MHS for forecasters at IMD. This report describes the various software tools used to make this ADPC and subsequent geospatial visualization.

1. Introduction

The real time data acquisition (whether satellite or ground based observation systems) for operational numerical weather prediction (NWP) centres is very crucial to monitor the atmospheric phenomena and to create initial conditions for the NWP models. The Direct Broadcast Network (DBNet) project was proposed by World Meteorological Organization (WMO), with an aim to provide near real-time access to the data from low earth orbit (LEO) satellites globally or regionally[1]. Most of the LEO meteorological satellites have the capability of direct broadcast, and a network of ground stations are established to receive the direct broadcast data stream. Each local ground station within the network receives data in real-time, but the coverage is limited to portion of orbit within the area of visibility of the local station (WMO, 2017).

In India, the Indian National Centre for Ocean Information Services (INCOIS, Hyderabad) has setup a ground station to receive the direct broadcast data from National Oceanic and Atmospheric Administration (NOAA)-15, NOAA-18, NOAA-19, MetOp-A and MetOp-B satellites. A regional processing centre is responsible for monitoring the reception of datasets from LEO satellites and the performance of DBNet of their region, particularly data relay, timelines. National Centre for Medium Range Weather Forecasting (NCMRWF), receives these satellite datasets in raw format (level-0) from INCOIS. These datasets are processed to level-1c or level-1d and being used for generating the spatial maps and used as observations in data assimilation (DA) system. This report briefly describes the dataset details in Section 2, methodology in Section 3, data processing in Section 4, data visualization in Section 5 and data monitoring in Section 6.

2. Data

The ground station at INCOIS receives the data from NOAA and MetOp satellites in high-resolution picture transmission (HRPT) and advanced high-resolution picture transmission (AHRPT) formats, respectively. The latter format is from the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT).

[1] https://community.wmo.int/activity-areas/wmo-space-programme-wsp/dbnet

These satellites cover India and neighbouring regions twice (two swaths) a day during ascending and descending orbits. The onboard sensors are: Advanced Very High-Resolution Radiometer/3 (AVHRR/3; for brevity referred as AVHRR from hereon), Advanced Microwave Sounding Unit-A (AMSU-A), Microwave Humidity Sounder (MHS) and High-Resolution Infrared Radiation Sounder/4 (HIRS/4; for brevity referred as HIRS from hereon). The three sensors i.e., AMSU-A, MHS and HIRS together form the Advanced TIROS (Television Infrared Observation Satellite) Operational Vertical Sounder (ATOVS). Presently, MHS sensor is not operational on NOAA-15 and NOAA-18. The processed data is used in the various DA systems at NCMRWF.

NCMRWF also receives data in AHRPT (EUMETSAT, 2019) format from MetOp-A and MetOp-B satellites and the onboard sensors are: AVHRR, AMSU-A, HIRS, IASI and MHS. AHRPT data is disseminated in L-band (Primary frequency at 1701.3 MHz, and a backup frequency of 1707.0 MHz) with data rate of 3.5 Mbps with local coverage radius up to 1500 km centring the receiving station. This data is distributed as a stream containing Channel Access Data Units (CADU) and require further processing to generate MetOp level-0 products. The stream holds multiplexed data from all MetOp instruments and spacecraft telemetry and administrative messages. Along with time ordered, frame synched and randomized, CADU's also hold Reed-Solomon error-correction decoding information and quality information. A CADU packet can be processed further up to Coded Virtual Channel Data Unit (CVCDU), Virtual Channel Data Unit (VCDU), MetOpizer Product Dissemination Units (M-PDU) and Instrument Source Packets as Consultative Committee for Space Data Systems (CCSDS) source packets. The details of MetOp data processing is given in Section 4.

3. Methodology

This section briefly describes the processing, visualizing, and monitoring of HRPT data from NOAA and AHRPT from MetOp satellites at NCMRWF. Two software programmes from EUMETSAT have been adopted for processing the data. They are: AAPP ((Atkinson, 2017); (Atkinson, 2020) & (Atkinson, 2019)) and MetOpizer (EUMETSAT, 2015). A brief description of these software are given below.

3.1 AAPP

The ATOVS and AVHRR Pre-processing Package (AAPP) developed and maintained by EUMETSAT's Satellite Application Facility for Numerical Weather Prediction (NWP SAF) since late 1990s. AAPP has been used for processing the direct readout data from NOAA-15 and its successors. AAPP has been upgraded after adding MetOp in 2006, Suomi-NPP in 2012 etc. AAPP mainly process the data from polar-orbiting satellites like NOAA series. AAPP has many tools to process the data at level-0 and level-1 i.e. generates the basic radiometric quantities such as brightness temperatures acquired along the satellite swath. AAPP also generates the cloud mask and sea surface temperature but does not retrieve the atmospheric temperature/humidity.

AAPP can handle various satellite sensors and has the capability to generate various data products at different processing levels. This is summarized in Table 1.

Table 1: Summary of AAPP capabilities

Sl. No.	Satellite/Platform	Sensors	Functionalities & Remarks
1	TIROS-N	HIRS	
		MSU	
		AVHRR/2	
2	NOAA-KLM	HIRS/3	• Decommutation of the instrument data from the HRPT data frame • Extraction of the calibration information • Navigation of the data • Calibration of the data • Pre-processing of the data (cloud mask included) • Mapping of the data of the sounder instruments on a common instrument grid (HIRS, AMSU, MHS or IASI) • Deriving a set of statistical parameters from the AVHRR data in the HIRS FOVs
		AMSU-A	
		AMSU-B	
		AVHRR/3	
3	NOAA-NN'	HIRS/4	
		AMSU-A	
		MHS	
		AVHRR/3	
4	MetOp	HIRS/4	• For MetOp, AAPP ingests "EPS Level 0" files (one file per instrument), then performs navigation, calibration and pre-processing as above. • In the case of IASI, level 1 processing is performed by the OPS-LRS (Operational Software – Local Reception Station), which is an optional extension to AAPP and is based on the OPS software provided to EUMETSAT by CNES.
		AMSU-A	
		MHS	
		AVHRR/3	
		IASI	

5	NPP	ATMS	• For NPP, the starting points for AAPP are Sensor Data Record files (SDR). For direct readout, you will need an external program, such as University of Wisconsin's CSPP package, or NASA's IPOPP package, to generate the SDR files.
		CrIS	
		VIIRS	
6	FY3	MWTS and MWTS-2	• The above same applies to FY-3, for which the China Meteorological Administration provide a processing package for direct readout use.
		MWHS and MWHS-2	
		IRAS	
		MWRI	

In terms of processing levels, AAPP defines the following data processing convention:

a) Raw data: Data as transmitted by the spacecraft, usually in sensor digital counts.
b) Level-0: Pre-processing steps like frame synchronisation have been performed on the raw data; in the case of MetOp, data from each instrument/sensor are separated into different files.
c) Level-1a: Raw counts of each instrument/sensor in separate file.
d) Level-1b: Geo-referenced and calibrated data (reversible: calibration coefficients are separated from raw data).
e) Level-1c: Geo-referenced and calibrated brightness temperatures and albedo (non-reversible: calibration coefficients are applied to numerical data). In the case of IASI, the spectra are apodized. In the case of microwave sounders, an antenna pattern correction will usually have been applied. This is the first geophysical product level.
f) Level-1d: Mapped and filtered data. Several instruments may be mapped to a common instrument grid. A cloud mask, or other derived products, may be included.

AAPP needs orbital elements for navigation. The bulletins provide the satellite position and velocity at any given time hence the software predicts the future positions. Often two-line elements (TLE) are used and being downloaded at NCMRWF at 6-hour interval. Alternatively, TLE data from NOAA and MetOp satellites from the MetOp Multi Mission Administration Message (MMAM) and can be obtained from HKTM file or downloaded from EUMETSAT. More details of data processing are provided in Atkinson, 2020. At NCMRWF AAPP version 8.5 has been installed on MIHIR High Performance Computing System (HPCS) to process the data as described by (Atkinson, 2019).

3.2 MetOpizer

The MetOpizer is a software toolkit from EUMETSAT to handle raw AHRPT data from MetOp satellites. MetOpizer is collection of programs as modules to perform different tasks on input data. There are many executables (modules) available in MetOpizer. More details of MetOpizer are given in (EUMETSAT, 2015). At NCMRWF, MetOpizer, version 3.51.1 has been installed on the MIHIR HPCS. Some of the MetOpizer functionalities are summarized in Table 2.

Table 2: Functionalities of MetOpizer

Sl. No	Module name	Purpose
1	*amsua_to_ccsds*	For converting NOAA L1b format AMSU-A product into a stream of CCSDS packets, or a series of images.
2	*amsub_to_mhs_ccsds*	For converting NOAA L1b format AMSU-B product into a stream of MHS format CCSDS packets, or to a series of images.
3	*avhrr_to_ccsds*	For converting NOAA AVHRR Level 1b file in either A stream of CCSDS packets either one per file or concatenated together, or a set of images.
4	*buffer_corrupt_filter*	Enables the modification/corruption of any input file/buffer with a single output file consisting of the input files and any specified corruptions.
5	*cadu_to_ccsds*	For converting one or more files in the PDA-PDU format (a concatenated list of CADU packets) into CCSDS packets. Takes input as list of input files containing CADU packets and outputs the one or more files containing lists of CCSDS packets.
6	*ccsds_to_l0*	For converting a stream of ISPs to an EPS L0 product. Takes input one or more streams of CCSDS packets (optionally previously generated MMAM files) outputs one EPS L0 product.

AAPP takes “EPS level 0” format data as input. But some reception systems deliver the data in CADU format (another raw format). This data need conversion to AAPP input data format. EUMETSAT developed MetOpizer software tool for this conversion. It is primarily being used to preprocess the raw data for AAPP. The main tools used in autonomous data processing chain are: *cadu_to_ccsds* and *ccsds_to_l0*.

4. Data Processing

This section briefly describes the processing of level-0 to level-1 data of NOAA and MetOp satellites in HRPT and AHRPT formats, respectively. The main submission script, ‘*submit_hrpt_process.sh*’ is provided in Annexure-I.

As a first step, the files received during the last 371 mins (6 hours and 11 mins) are created into a list (a simple text file) and passed on to the main submission script, '*submit_hrpt_process.sh*'. The main script calls a subscript based on the satellite, either NOAA ('*noaa_hrpt_process.sh*'; see Annexure-II) or MetOP ('*metop_hrpt_process.sh*'; see Annexure-III). The flow diagram of the data processing chain is shown in Figure 1. The following subsections further describe the processing steps.

4.1 NOAA HRPT Data

NCMRWF receives HRPT data from NOAA-15, NOAA-18, and NOAA-19 satellites. As mentioned earlier, on NOAA-15 and NOAA-18, the MHS sensor is faulty hence only AMSU-A, AVHRR and HIRS sensor data are processed. The raw data has been pushed by INCOIS in near real-time, and at NCMRWF, this is processed at every six hours. The data has been processed with AAPP from level-0 and the operational procedure is given in Annexure-II.

The input raw data needs pre-processing due to 348 extra bytes at the end of each frame. Hence a perl script, called '*strip_hrpt.pl*'[2] is used for removing these extra bytes of data for all NOAA satellites and processed with AAPP. AAPP's processing script, "*AAPP_RUN_NOAA*" is the main script called by "*noaa_hrpt_process.sh*" to process the data. AAPP has the capability to generate individual sensor data (level-1b, level-1c) and also able to map one sensor grid on to other sensors. For NWP purposes, level-1d data has been used (mapping AVHRR, MHS, on to HIRS grid). AAPP also converts these processed files into user friendly Hierarchical Data Format (HDF) files. The data processed by AAPP is plotted using PyTroll's *satpy* (v 0.21.0) (Raspaud, et al., 2020). The next section describes the data visualisation and the scripts involved. Similarly, the level-1c data products from AMSU-B, and MHS sensors data are plotted using a separate python script as described in the next section.

4.2 MetOp AHRPT Data

MetOp-A and MetOp-B data are received in another raw data format called CADU. Using EUMETSAT's MetOpizer tool, the raw data is converted into level-0 format. The main MetOpizer tools used for data pre-processing are: *cadu_to_ccsds* and *ccsds_to_l0*.

An issue with MetOp-B's AMSU-A data was flagged, when the MetOpizer module '*ccsds_to_l0*' failed. A simple error diagnosis indicated a possible data corruption at the start of the file, which could be due to the fact that data reception centre starts acquiring the data at low elevations. Generally, the recommended data reception is 5° above horizon. Hence first 1000 frames (based on ephemeris and attitude information) from each file were stripped. Each frame contains 1024 bytes of data; hence 1024000 bytes of data has been removed from MetOp-B's AMSU-A sensor. No such issues were encountered with MetOp-A. The shell script, '*metop_hrpt_process.sh*' (see Annexure-III) is used to process the data from sensors onboard MetOp satellites (i.e. AMSU-A, AVHRR, MHS and HIRS). After processing these data for

[2] This script '*strip_hrpt.pl*' was provided Dr. Nigel Atkinson, UK Met Office.

individual sensors, these files are converted to HDF. Finally, the MetOp satellite data (AVHRR, AMSU-A, MHS) is visualised using Python and wrapper shell scripts, which is described in the next section.

5. Data Visualization

This section briefly describes the visualization of processed HRPT/AHRPT data from NOAA and MetOp satellites for both level-1b and level-1c data products. There are several options available for visualisation of the data for more information about various tools please refer to https://www.nwpsaf.eu/site/software/aapp/visualisation/. In this technical report, the existing python-based framework of PyTroll at NCMRWF has been leveraged for visualisation.

To plot level-1b data, a wrapper shell scrip called '*run_l1b_plot.sh*' (see Annexure-IV) is invoked which sets up the appropriate Python environment, creates the list of level-1b files to be plotted and generates temporary Python scripts in system 'scratch' area[3], which are based on a template Python script called '*plot_hrpt_l1b_mas.py*' (see Annexure-VI) and makes use of the *satpy* reader[4] called '*avhrr_l1b_aapp*'. The satellite imageries are generated for AVHRR channels: 1, 2, 4 and 5 centred at wavelengths: 0.630 µm, 0.862 µm, 10.8 µm and 12.00 µm, respectively. The channels 1 and 2 are in visible range whereas channels 4 and 5 are in thermal infrared range. The Figure 2 is representative snapshot of the Tropical Cyclone (TC) AMPHAN on 18 May 2020, wherein the top panels are in the visible range (i.e. channels 1 and 2), while the bottom panels show the TC in infrared channels as observed by AVHRR sensor onboard NOAA-18 satellite. The Figure 3 through Figure 5 provide a progressive snapshot of the TC AMPHAN as captured by AVHRR sensor onboard the three satellites: NOAA-18, NOAA-19 and MetOp-A, respectively, at 12µm wavelength (Channel-5). The dates of progression in each of the above three Figures correspond to May 18th, May 19th and May 20th, 2020.

A similar approach to AVHRR has been adopted to visualize level-1c data from AMSU-A and MHS sensors, wherein a wrapper script called '*run_l1c_plot.sh*' (see Annexure-V) is the governing script, however it does not create Python scripts based on a template rather it makes use of a single script called '*aapp1c_plot.py*' developed by EUMETSAT and further modified by the third author of this report to include matplotlib's *basemap* projection for the Indian domain. This version of '*aapp1c_ploy.py*' is presented as Annexure-VII. The wrapper script as usual provides the Python environment and arguments to the above-mentioned Python script.

These images were generated after processing the data for 15 channels for AMSU-A and five channels for MHS. Figure 6 shows the MHS imagery (from NOAA-19 satellite) for TC AMPHAN on 18 May 2020. These five images of TC AMPHAN are as acquired by MHS at microwave

[3] The automatically generated Python scripts can be found on MIHIR HPCS at: /scratch/satviz/tmp/aapp/plt_l1b/ folder. This area is cleaned as per MIHIR data policy.

[4] The Python class definition is available at: https://satpy.readthedocs.io/en/latest/api/satpy.readers.html?highlight=aapp#satpy.readers.aapp_l1b.AVHRRAAPPL1BFile

channels centred at: 89.0V, 157.0V 183 (±3.0) H, 183 (±3.0) H and 190.3V with bandwidths of 2.8, 2,8, 2.0, 1.0 and 2.0 GHz respectively. The H and V stand for horizontal and vertical polarization. The spatial resolution of this passive microwave sensor, MHS, works out to be 16 km at centre.

The Figure 7 highlights the progression of TC AMPHAN as seen through a passive microwave sensor, i.e. NOAA-19 MHS. The horizontal direction (row-wise) span through the five microwave spectral channels, while the vertical (column-wise) depict the progression of the TC, with the last row indicating the landfall of AMPHAN on May 20th, 2020. Similarly, the Figure 8 depicts four channels of MHS sensor from MetOp-A satellite for dates May 16th through 18th. The Figure 9 highlights the progression (vertically or row-wise) of TC AMPHAN by a sounder called AMSU-A onboard three different platforms: NOAA-18, NOAA-19 and MetOp-A (horizontally or column wise). Note that AMSU has an effective spatial resolution of 48 km at centre. Finally, the Figure 10 shows the life cycle of TC AMPHAN (from May 12th to May 20th, 2020) using a single thermal band (channel-5) of AVHRR sensor onboard both the NOAA and MetOp series of satellites.

As highlighted by Figure 2 through Figure 10, a visualisation platform can provide a very good synoptic understanding for nowcasting purposes. Along with the visualisation, the above-mentioned ADPC will provide the necessary, quality-controlled, rightly-formatted and timely data to NWP's DA system for ingestion and assimilation.

6. Data Monitoring

Any automatic data processing chain is incomplete if it does not have a monitoring and alerting system that can alert any break in work flow right from data reception to product generation. The file count varies daily due to orbit shift; and on average (per day basis) NCMRWF receives 16 files or more making up to 2.5 GB of total size. Given the quantum of data, a checksum procedure is done on the size and number of files on daily and monthly time scales. The data monitoring script, '*plt_delay.R*', was written in R language (see Annexure IX) and is controlled by a wrapper script, '*run_plot_time_delays.sh*', shown in Annexure-VIII). The monitoring is done every day at 11AM and plots of previous day data with time delay between data reception at INCOIS (extracted based on file name) and data reception at NCMRWF (local time). The time difference between these two times is plotted to monitor for each file and is depicted in Figure 11, valid for 15 April, 15 May and 15 June.

Similarly, monthly data monitoring is done on 5th of every month for the previous month. The Figure 12 depicts monthly monitoring valid for April, May and June, 2020. As seen in Figure 12, reception delays in the order of 2-3 days were observed in the months of April and May (red ellipses), especially during the weekends, which have been addressed with INCOIS. The data monitoring aspect helps to understand the time delays in data reception at NCMRWF, if the delay is more than specified time, a follow up action is initiated.

7. Acknowledgements

We acknowledge the Head, NCMRWF for his continuous support. We also thankful to D.G., India Meteorological Department for taking initiative for supplying the data at NCMRWF. We also thankful to INCOIS, Hyderabad for sending the data to NCMRWF. We also thank Dr. Nigel Atkinson, UK Met Office for providing us the perl script and troubleshooting data issues.

8. References

[1]. Atkinson, N. (2017). *ATOVS and AVHRR Pre-processing Package (AAPP): Overview.* EUMETSAT, NWP SAF. Darmstadt: EUMETSAT. Retrieved 07 19, 2020, from https://nwp-saf.eumetsat.int/site/download/documentation/aapp/NWPSAF-MO-UD-004_Overview_v8.0.pdf

[2]. Atkinson, N. (2019). *ATOVS and AVHRR Pre-processing Package (AAPP): Installation Guide.* EUMETSAT, NWP SAF. Darmstadt: EUMETSAT. Retrieved 07 14, 2020, from https://nwp-saf.eumetsat.int/site/software/aapp/documentation/aapp-v8-installation-guide/

[3]. Atkinson, N. (2020). *ATOVS and AVHRR Pre-processing Package (AAPP): User Guide, Ver 8.5.* EUMETSAT, NWP SAF. Darmstadt: EUMETSAT. Retrieved 07 18, 2020, from https://nwp-saf.eumetsat.int/site/software/aapp/documentation/aapp-v8-userguide/

[4]. EUMETSAT. (2015). *MetOpizer User Guide, v4A.* EUMETSAT. Darmstadt: EUMETSAT. Retrieved 07 20, 2020, from https://www.eumetsat.int/website/wcm/idc/idcplg?IdcService=GET_FILE&dDocName=PDF_METOPIZER_USERGUIDE&RevisionSelectionMethod=LatestReleased&Rendition=Web

[5]. EUMETSAT. (2019). *TD 18 MetOp Direct Readout AHRPT Technical Description.* EUMETSAT. Darmstadt: EUMETSAT. Retrieved 07 20, 2020, from http://www.eumetsat.int/website/wcm/idc/idcplg?IdcService=GET_FILE&dDocName=PDF_TD18_METOP_A_DIRECT_READ&RevisionSelectionMethod=LatestReleased&Rendition=Web

[6]. Raspaud, M., Hoese, D., Lahtinen, P., Dybbroe, A., Finkensieper, S., Proud, S., . . . Sigurðs, E. (2020). *pytroll/satpy: Version 0.21.0.* Zenodo. doi:10.5281/zenodo.3742294

[7]. WMO. (2017). *Guide to the Direct Broadcast Network for Nearr-real-time Relay of Low Earth Orbit Satellite Data: Attachment to the Guide to the WMO Information System (WMO-No. 1061).* WMO. Geneva: WMO. Retrieved 07 17, 2020, from https://library.wmo.int/doc_num.php?explnum_id=4135

9. Figures

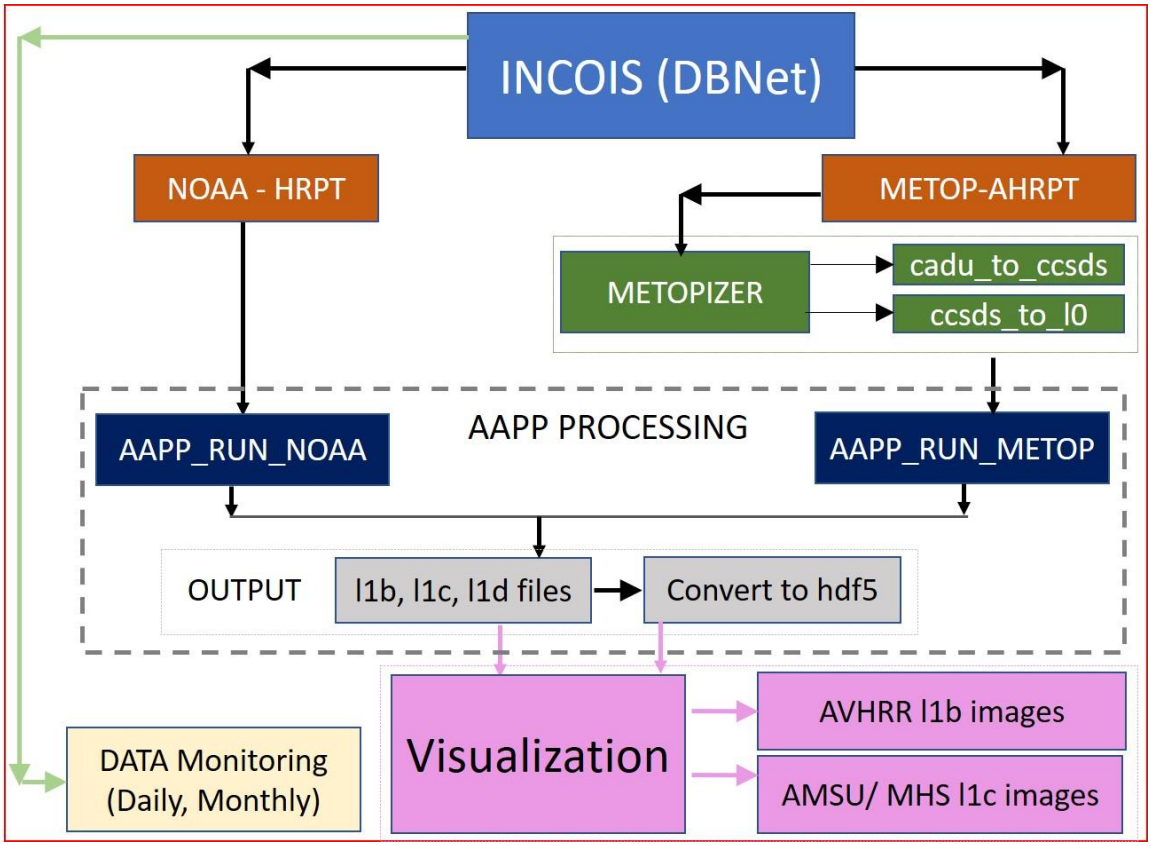

Figure 1: The flowchart of data processing at NCMRWF for NOAA and MetOp satellites.

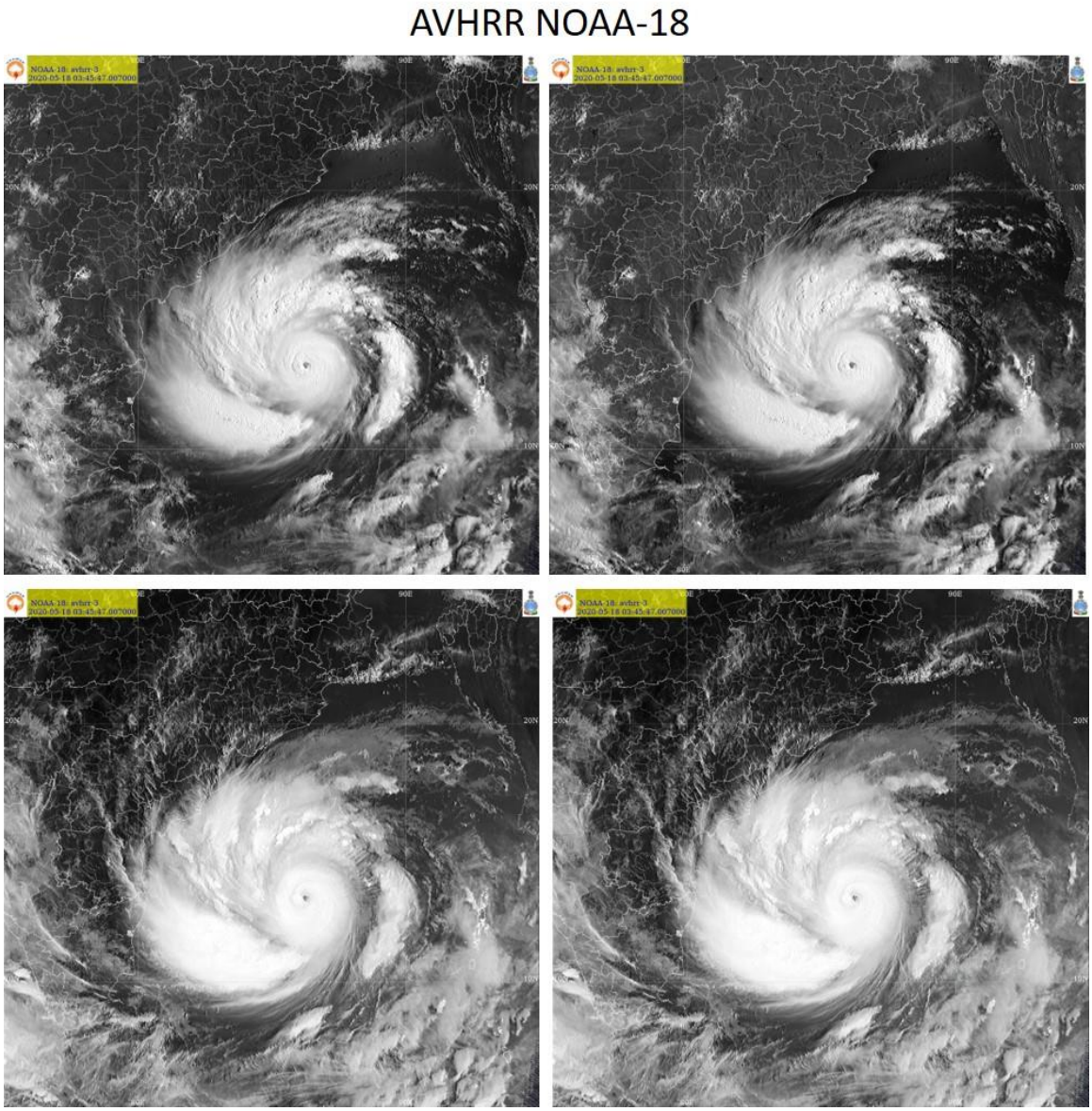

Figure 2: NOAA-18 AVHRR channels: 1 & 2 (top panel), channels: 4 & 5 (bottom panel) of TC AMPHAN valid on 18 May 2020.

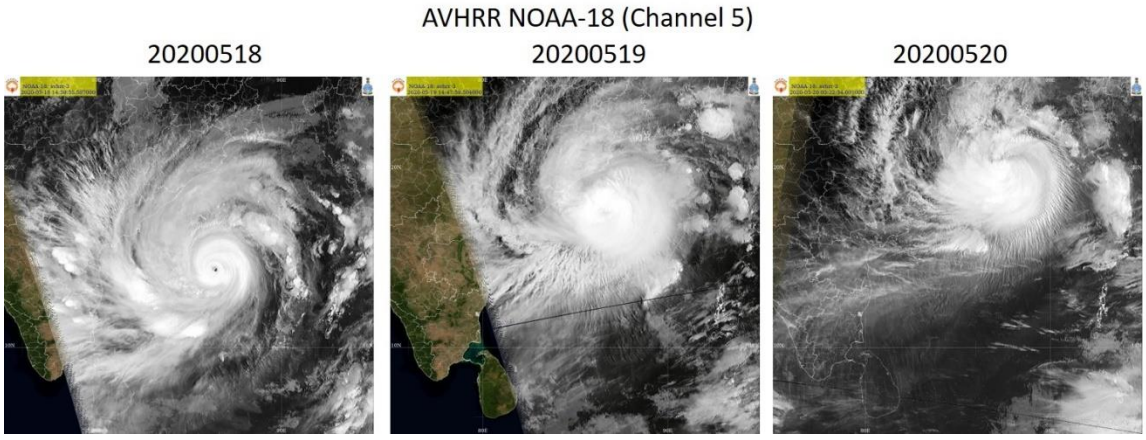

Figure 3: NOAA-18 AVHRR channel-5 imagery from NOAA-18 valid on 18, 19, 20 May 2020, highlighting the TC AMPHAN progression.

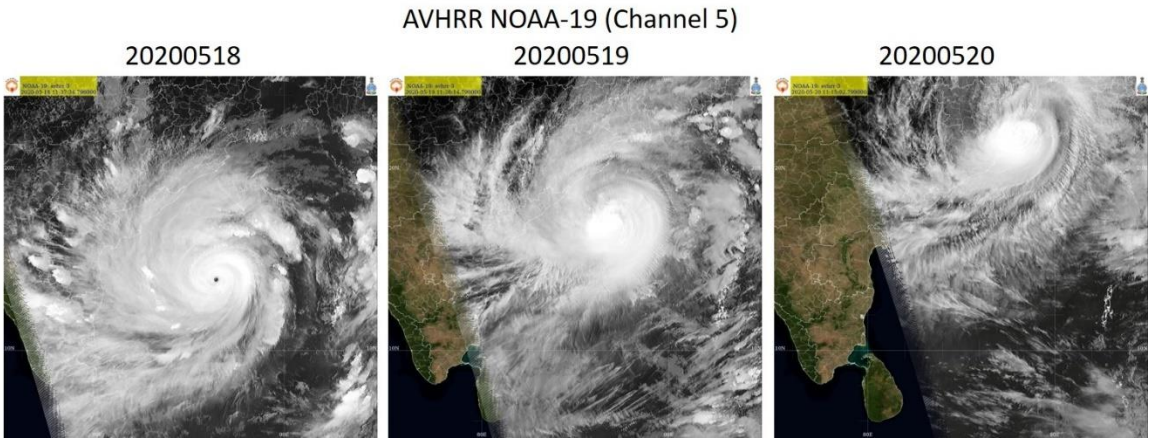

Figure 4: NOAA-19 AVHRR channel-5 imagery from NOAA-18 valid on 18, 19, 20 May 2020, highlighting the TC AMPHAN progression.

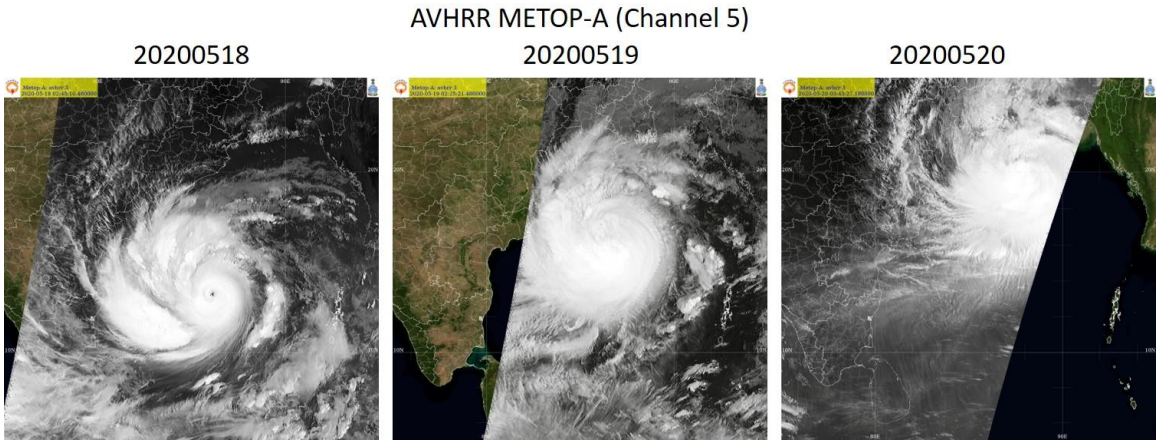

Figure 5: METOP-A AVHRR channel-5 imagery from NOAA-18 valid on 18, 19, 20 May 2020, highlighting the TC AMPHAN progression.

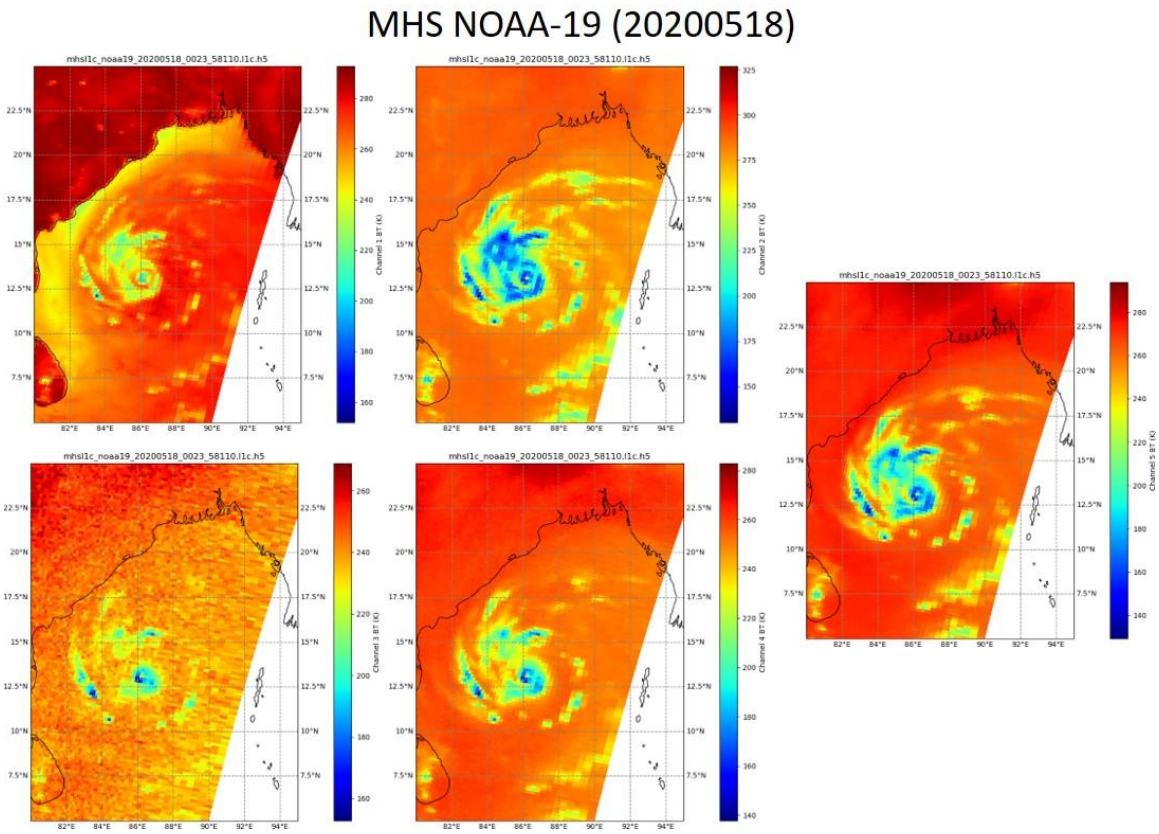

Figure 6: A spectral snapshot of TC AMPHAN as observed by MHS onboard NOAA-19 at a spatial resolution of 16 km and valid on 18 May 2020.

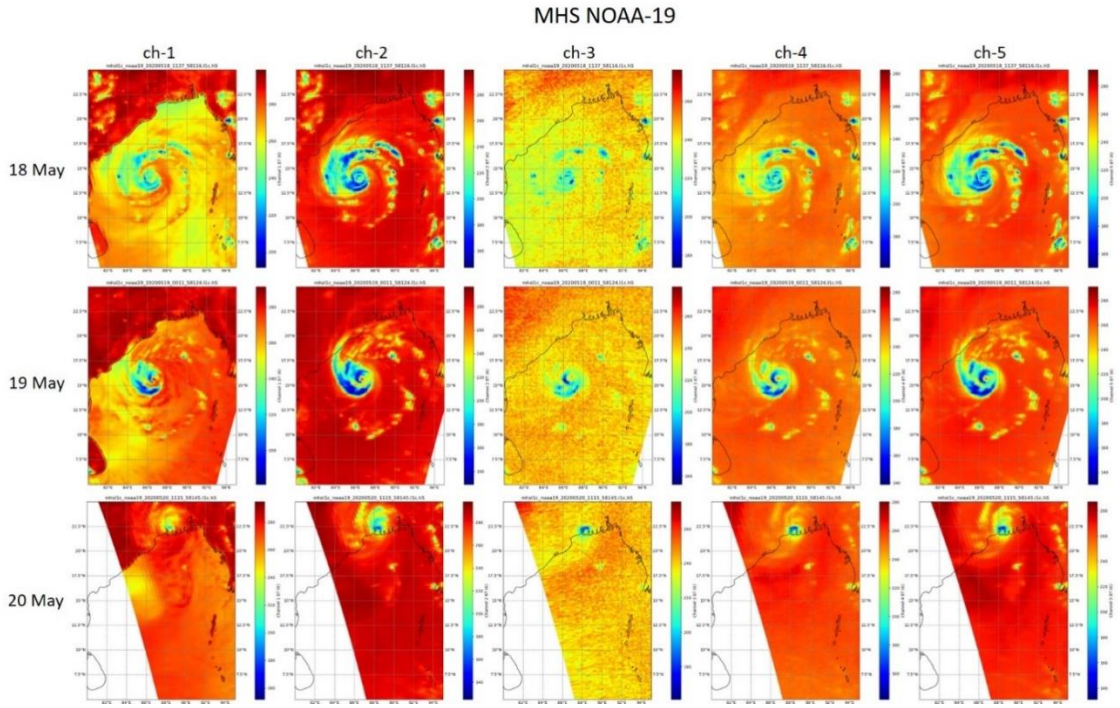

Figure 7: Progression of TC AMPHAN from 18 to 20 May, 2020 as observed by NOAA-19 MHS.

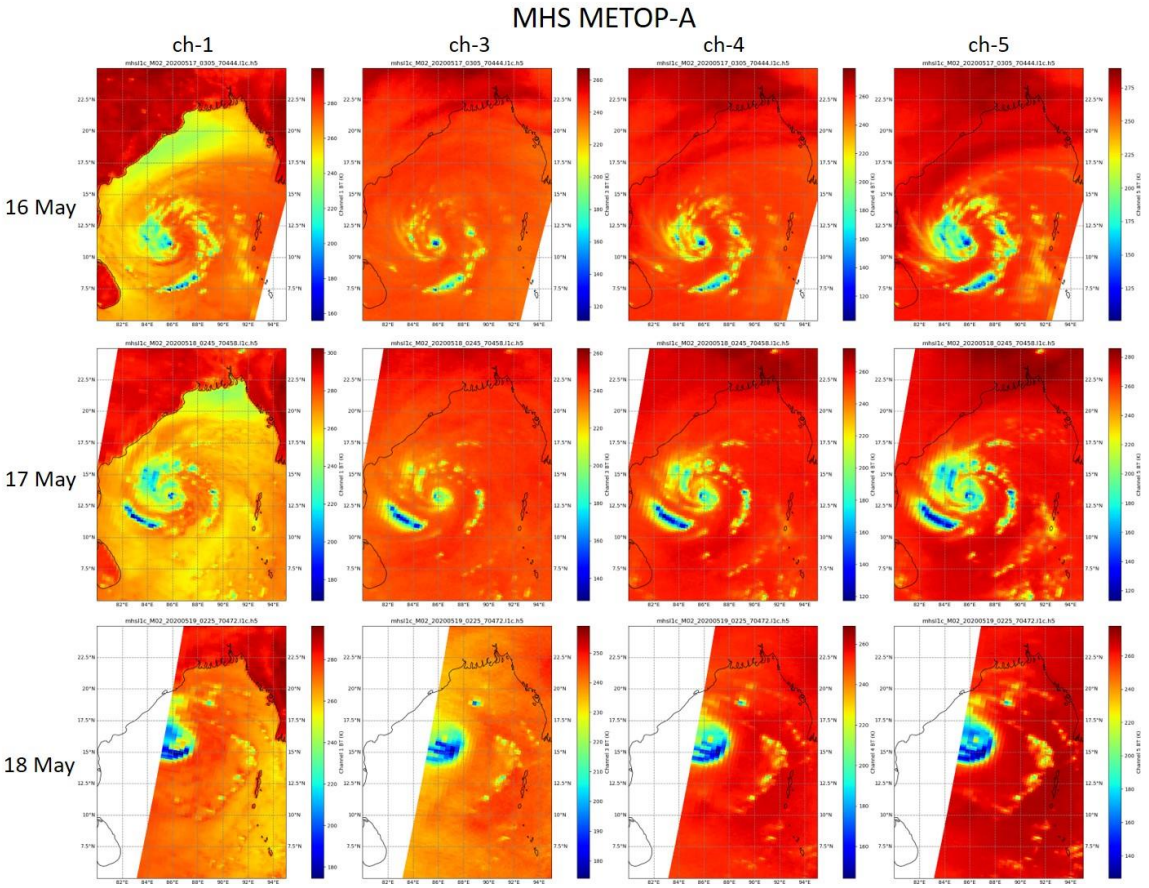

Figure 8: Progression of TC AMPHAN from 16 to18 May, 2020 as observed by METOP-A MHS.

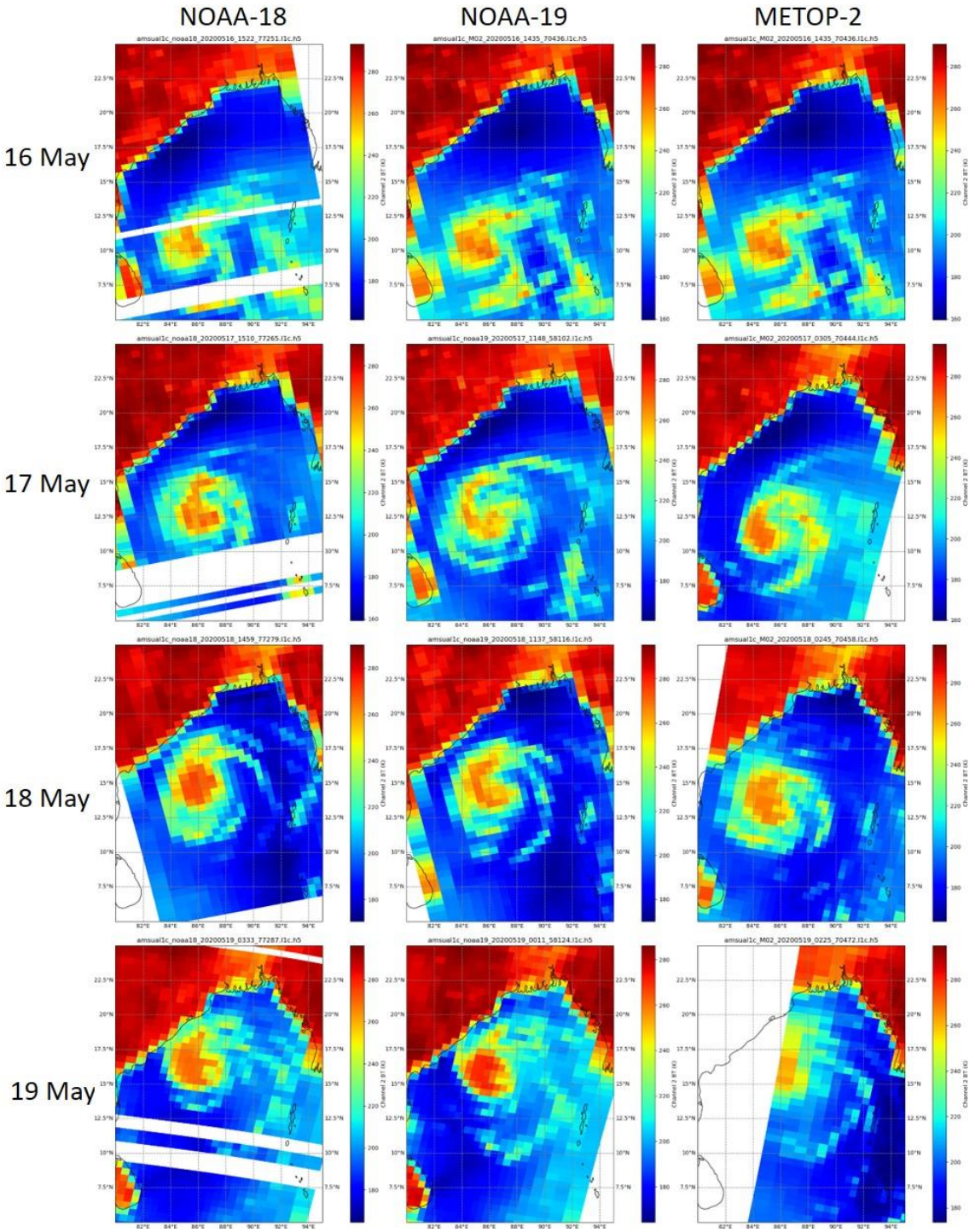

Figure 9: The AMSU-A channel-2 imagery from NOAA-18, NOAA-19 and METOP-A valid on 16, 17, 18 and 19 May 2020.

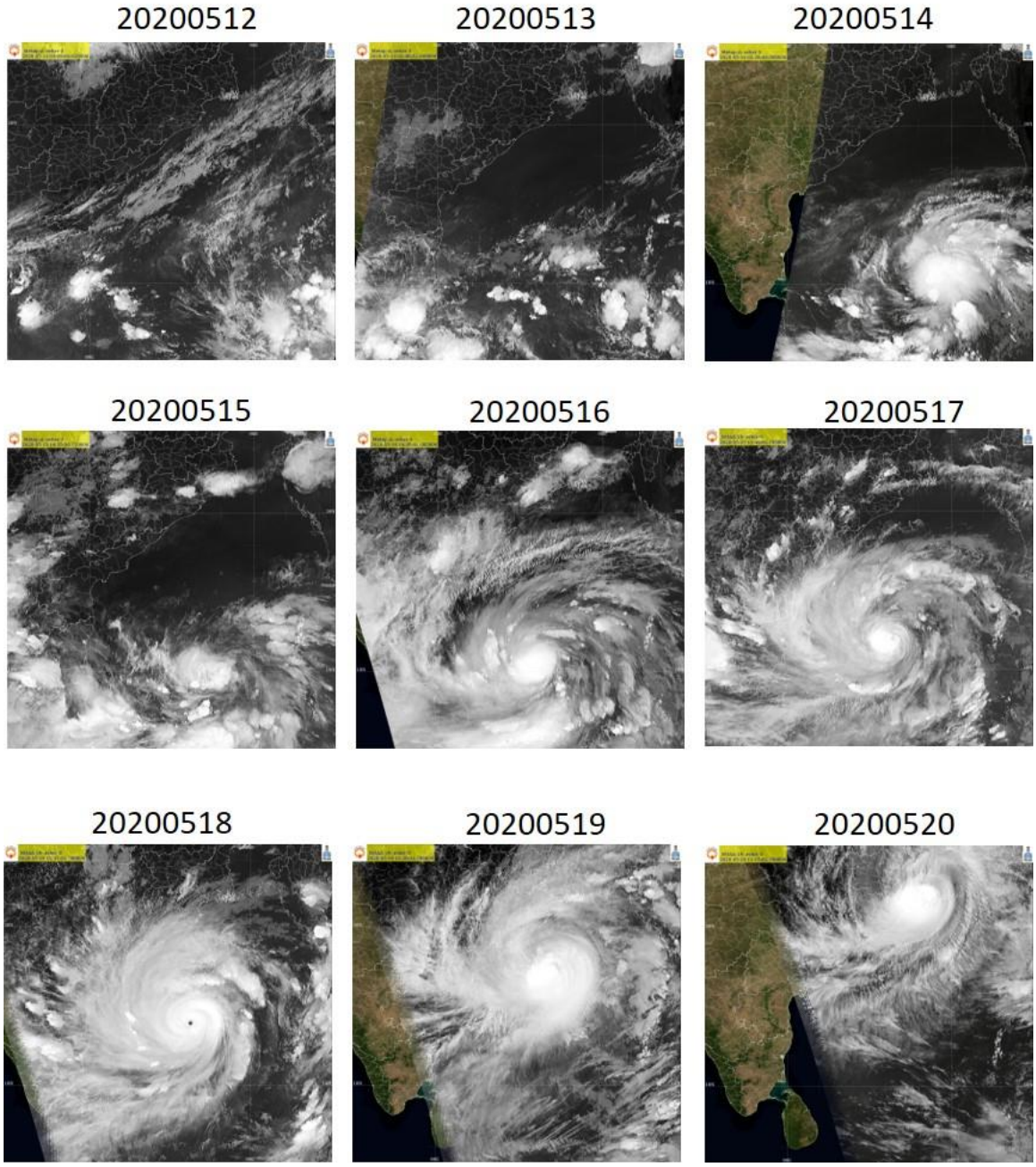

Figure 10: The cyclogenesis of TC AMPHAN as observed by AVHRR (Channel-5) onboard NOAA and METOP series of satellites, valid from 12-20 May 2020.

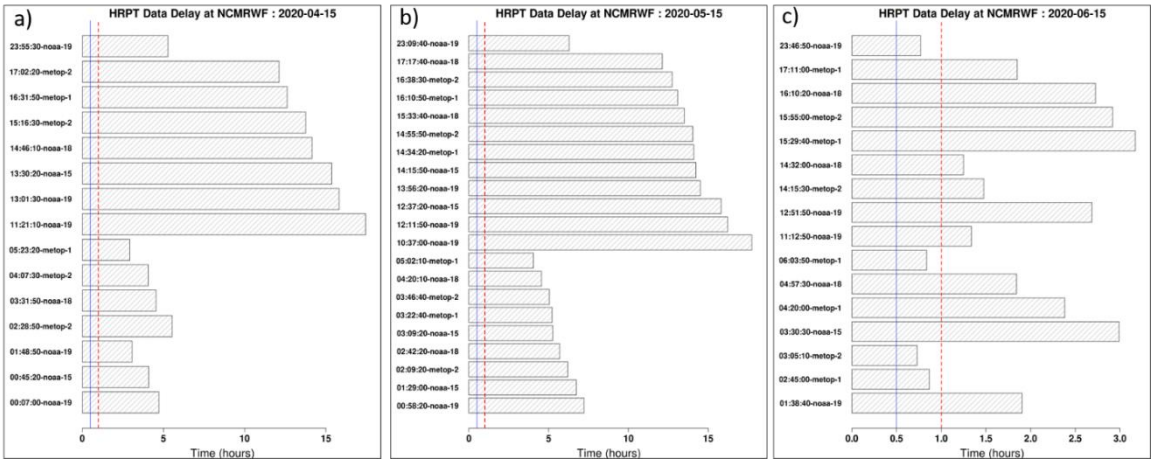

Figure 11: Data monitoring of HRPT/AHRPT reception at NCMRWF from INCOIS valid for 15 April, 15 May and 15 June 2020.

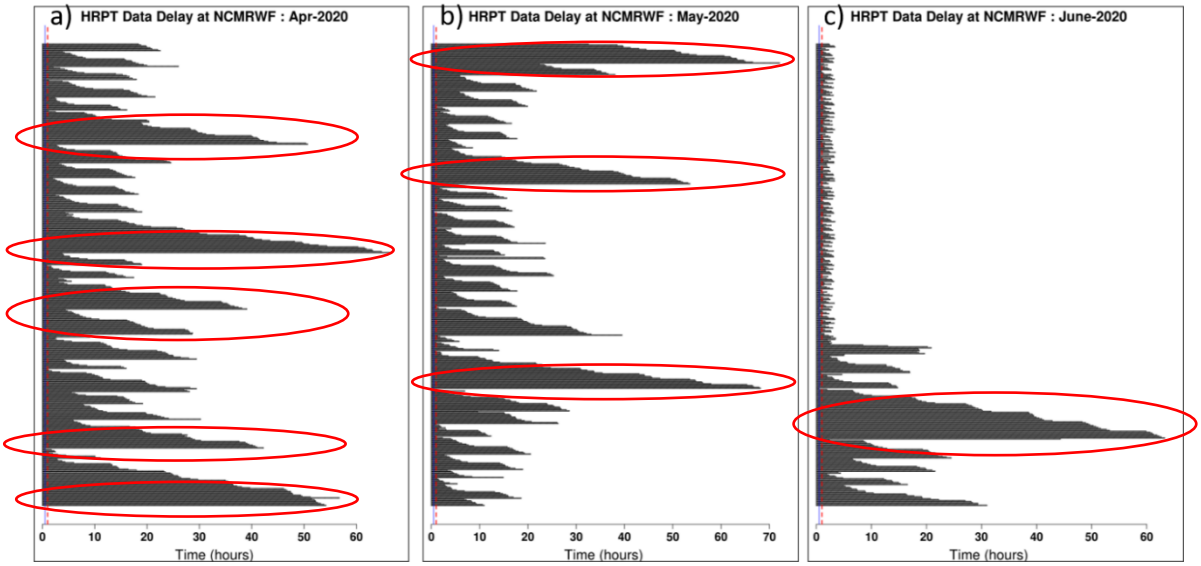

Figure 12: Monthly data monitoring of HRPT/AHRPT reception at NCMRWF from INCOIS valid for months: April, May and June 2020.

10. ANNEXURES

Annexure-I: submit_hrpt_process.sh

```
#!/bin/bash
## Get the input hrpt data as list and pass for data processing ....
set -x

export HOMED=/home/satviz/hrpt_noaa/processing_hrpt_data
export SCRP=$HOMED/scripts
export SRC=$HOMED/src
export DATA=/home/hrpt_incois
export LOG=/scratch/satviz/tmp/aapp/LOGS
export AAPP_PREFIX=/home/satviz/AAPP/AAPP_8.5U/AAPPU

find $DATA/2* -cmin -371  >list.hrpt

for file in `cat list.hrpt` ; do
   # Pass on the data file to process corresponding script (NOAA-
15/18/19 / MetOp)
pfile=`echo $file | awk -F "/" '{print $NF}'`
  sensor=`echo $file | awk -F "." '{print $NF}'`
  sat=`echo $sensor | cut -c1-4`

  if [[ $sat  ==  "noaa" ]] ; then
     echo "INPUT DATA from NOAA  Satellite -- $sat - $sensor"
     $SCRP/noaa_hrpt_process.sh  $file  $sensor  $sat  >&
$LOG/log.noaa_${pfile}  &

elif [[ $sat == "meto" ]] ; then
     echo "INPUT DATA from MetOp Satellite -- $sat - $sensor"
     $SCRP/metop_hrpt_process.sh   $file  $sensor  $sat  >&
$LOG/log.metop_${pfile}  &

  else
     echo " Input Data neither NOAA nor MetOp "
  fi
   echo "file - $sensor -- $sat"

 sleep 60
done # file

set +x
```

Annexure-II: noaa_hrpt_process.sh

```
#!/bin/bash
# 1. Processing NOAA-Level 0 HRPT data to level-1d (HIRS Grid)
# 2. Processing NOAA-AVHRR Data into level 1b, and level 1c

set -x
export pid=$$
export HOMED=/home/satviz/hrpt_noaa/processing_hrpt_data
export SCRP=$HOMED/scripts
export SRC=$HOMED/src
export OUTDIR=/home/satviz/hrpt_noaa/processing_hrpt_data/data
export AAPP_PREFIX=/home/satviz/AAPP/AAPP_8.5U/AAPPU

infile=$1
sensor=$3
sat=$2

source $SRC/remodule_process_metop

[ $AAPP_PREFIX ] || { echo "AAPP_PREFIX is not defined"; exit; }
[ -d "$AAPP_PREFIX" ] || { echo "AAPP_PREFIX is not a directory"; exit;
}

. "$AAPP_PREFIX/ATOVS_CONF"               #AAPP environment

ofile=`echo $infile | awk -F "/" '{print $NF}'`

odate=`echo $ofile | cut -c1-9`
export WRK=/scratch/${LOGNAME}/tmp/aapp/noaa/$odate/${pid}
export ODIR=$OUTDIR/$odate
export level1=$WRK/level1

rm -rf $WRK
mkdir -p $level1 $ODIR $WRK; cd $WRK
# ------------------------------------------------------------------
#Processing NOAA Level 0 Sounding data and creating level 1d (hirs
#grid) ------------

## Pre-Processing raw data (For the NOAA-15/18/19 data, we have 348
extra bytes at the end of each frame,
#  i.e.  22528 bytes instead of 22180). We have to remove it first
#!/usr/bin/perl   $SRC/strip_hrpt.pl <$infile> ./$ofile
usleep 300
```

```
if [[ $sat == "noaa-15" || $sat == "noaa-18" ]] ; then
instr="AMSU-A HIRS AVHRR"
elif [[ $sat == "noaa-19" ]] ; then
instr="AMSU-A MHS HIRS AVHRR"
else
  echo "The satellite is not NOAA-15/18/19 .. exiting now ... "
  exit 1
fi

grids="HIRS"
## Running AAPP
time AAPP_RUN_NOAA -i "$instr" -g "$grids" -o $WRK $ofile>
log_noaa_1.out 2>&1

status=$?
if [ $status != 0 ]; then
  echo "AAPP_RUN_NOAA failed"
  exit 1
else
outfiles=$(ls *.l1b *.l1c *.l1d 2>/dev/null)
  for file in $outfiles; do
    if [ -s $file ]; then
      mv $file $level1/
    else
      rm $file
    fi
  done # file

## checking whether level-1 files created
  if [ "$outfiles" ]; then
    echo "AAPP_RUN_NOAA ended: output in $level1"
  else
    echo "No output files were generated"
  fi
fi

## Converting to HDF5 files
cd $level1
has_hdf5=Y
for file in $(ls hrpt*.l1b amsu*.l1c mhs*.l1c hirs*.l1c hirs*.l1d); do
  if [ $has_hdf5=Y ]; then
    convert_to_hdf5 $file || has_hdf5=N
  fi
done # file
```

```
## checking whether converted to hdf5
if [ $has_hdf5 ]; then
  echo "Converted l1c files to hdf5"
else
  echo "Have not converted l1c files to hdf5"
fi

MAIA4_USE_GFS="no"
has_hdf5=Y
cd $level1

MAIA4_RUN_AVHRR $PWD/hrpt_*.l1b

status=$?
if [ $status != 0 ]; then
  echo "MAIA4_RUN_AVHRR failed"
#  exit
else
  convert_to_hdf5 ${ofile}.l1c has_hdf5=N
fi

if [ $has_hdf5 ]; then
  echo "Converted HRPT-l1c files to hdf5"
fi

# Moving all l1b, l1c, l1d files to ODIR
mv * $ODIR/
rm -rf $WRK

echo "Successful Creation of NOAA files .... "
exit
set +x
```

Annexure-III: metop_hrpt_process.sh

```
#!/bin/bash
# 1. MetOp-2 A/ 1 B raw CADU data to AAPP level 1c
# 2. Processing MetOp-2_A/1_B data to level-1d (HIRS Grid)

set -x
export pid=$$
export HOMED=/home/satviz/hrpt_noaa/processing_hrpt_data
export SCRP=$HOMED/scripts
export SRC=$HOMED/src
export OUTDIR=/home/satviz/hrpt_noaa/processing_hrpt_data/data
source $SRC/remodule_process_metop
export AAPP_PREFIX=/home/satviz/AAPP/AAPP_8.5U/AAPPU

source $SRC/remodule_process_metop

###_-----------------------------------------
export MetOpZ=/home/apps/SiteSoftwares/gnu/metopizer_3.51.1
export
PATH=$PATH:$MetOpZ/bin:/home/apps/SiteSoftwares/gnu/libjpeg_utility/bin:

export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/apps/SiteSoftwares/gnu/fec_3.0.1/
lib:/home/apps/SiteSoftwares/gnu/libjpeg_utility/lib:

infile=$1
sensor=$3
sat=$2
#_________________
#==== Check paths

cadu_to_ccsds>/dev/null 2>&1 || { echo "Error: metopizer function
cadu_to_ccsds not found: please modify your PATH"; exit 1; }

[ "$(ldconfig -p | grep libfec)" ] || [ "$(IFS=:; for dir in
$LD_LIBRARY_PATH; do [ -x $dir/libfec.so ] && echo Y; done)" ] || \
{ echo "Error: libfec needs to be either centrally installed or its
location added to LD_LIBRARY_PATH"; exit 1; }

[ $AAPP_PREFIX ] || { echo "AAPP_PREFIX is not defined"; exit; }
[ -d "$AAPP_PREFIX" ] || { echo "AAPP_PREFIX is not a directory"; exit;
}

. "$AAPP_PREFIX/ATOVS_CONF"               #AAPP environment
```

```
ofile=`echo $infile | awk -F "/" '{print $NF}'`

echo "INFILE = $infile  "
echo "OFILE  = $ofile  "
echo "SENSOR = $sensor "
echo "SATLTE = $sat "

odate=`echo $ofile | cut -c1-9`
export WRK=/scratch/${LOGNAME}/tmp/aapp/metop/$odate/${pid}
export ODIR=$OUTDIR/$odate
export level1=$WRK/level1
export level0=$WRK/level0
export metpzw=$WRK/metopizer_work
export
metpzconfig=/home/srinivas/sree/misc/AAPP_8.5U/test_cases/tars/operation
al/config_files

rm -rf $WRK
mkdir -p $ODIR $WRK $level0 $level1 $metpzw $metpzw/MMAM ; cd $WRK
###
#export DIR_NAVIGATION=$WRK/orbelems
#export DIR_DATA_TLE=$WRK/orbelems/tle_db
#mkdir -p $DIR_DATA_TLE $DIR_NAVIGATION/satpos
###
# ----------------------------------------------------------------
# 1. Processing MetOp-A/B Level 0 Sounding data and creating level 1d #
(hirs grid) ------------
#==== Set the satellite and input file (change these as needed)

# There seemed to be a data corruption near the start of the file that #
was causing AMSU-A processing in "ccsds_to_l0" to fail.
# As a  workaround I removed the  first 1000 frames (each frame is #1024
bytes) from the file, like this:

if [[ $sat == "metop-1" ]] ; then
instr="AMSU-A HIRS AVHRR"
   echo " Processing for MetOp-1 Satellite ... "
   tail -c +1024000  $infile> ${ofile}
bsat=M01
else
cp  $infile  ${ofile}
bsat=M02
fi # input_data_process
```

```
infile=$ofile

instruments_metopizer="amsuamhshirsavhrriasi"
instruments_AAPP="AMSU-A MHS HIRS AVHRR"

#==== Run the MetOpizer

cd metopizer_work

ccsdsfile=$(basename $infile)_ccsds

$MetOpZ/bin/cadu_to_ccsds --in ../$infile --out $ccsdsfile || { echo
"cadu_to_ccsds failed"; exit 1; }

instr=hktm
echo $instr
$MetOpZ/bin/ccsds_to_l0 --config-apid $metpzconfig/ccsds.config \
            --config-mphr $metpzconfig/mpr-config-${bsat} \
            -d $PWD -g MMAM -t tle -i ${ccsdsfile}.${instr}.ccsds || {
echo "ccsds_to_l0 failed"; exit 1; }

tlefile=$(ls MMAM/TLE*.tle | tail -1)       #e.g.
TLE_M02_002690_20160323010000Z_20160323160000Z.tle

[ -z $tlefile ]&& { echo "No tle file created"; exit 1; }

for instr in $instruments_metopizer; do
  echo $instr
  $MetOpZ/bin/ccsds_to_l0 --config-apid $metpzconfig/ccsds.config \
              --config-mphr $metpzconfig/mpr-config-${bsat} \
              -d $PWD -u MMAM -i ${ccsdsfile}.${instr}.ccsds || { echo
"ccsds_to_l0 failed"; exit 1; }
done

mv *HRP*Z $level0/

echo "======================="
which cadu_to_ccsds
echo "======================="
#==== Rename the TLE file for AAPP use

tlefile=$(ls MMAM/TLE*.tle | tail -1)
#TLE_M02_002690_20160323010000Z_20160323160000Z.tle
[ -z $tlefile ]&& { echo "No tle file created"; exit 1; }
```

```
set $(IFS="_"; echo $tlefile)
timestamp=$4

#==== Run AAPP to convert to level 1c
cd $WRK
#AAPP_RUN_MetOp -d $level0 -o $level1 -i "$instruments_AAPP" -g " "
AAPP_RUN_MetOp -d $level0 -o $level1 -i "$instruments_AAPP" -g "HIRS"

## Converting to HDF5 files
cd $level1
has_hdf5=Y
for file in $(ls *.l1b *.l1c *.l1d); do
  if [ $has_hdf5=Y ]; then
    convert_to_hdf5 $file || has_hdf5=N
  fi
done # file

## checking whether converted to hdf5
if [ $has_hdf5 ]; then
  echo "Converted l1b l1c l1d files to hdf5 and moved to .. \n ---------
$ODIR --------- "
# Moving all l1b, l1c, l1d files to ODIR
mv * $ODIR/
else
  echo "Have not converted l1c files to hdf5"
fi

##
# ---------------------- END - Processing 1 -------------------------

MAIA4_USE_GFS="no"
has_hdf5=Y
cd $level1

MAIA4_RUN_AVHRR $ODIR/hrpt_*.l1b

status=$?
if [ $status != 0 ]; then
  echo "MAIA4_RUN_AVHRR failed"
#  exit
else
  convert_to_hdf5 ${ofile}.l1c has_hdf5=N
fi

if [ $has_hdf5 ]; then
```

```
  echo "Converted HRPT-l1c files to hdf5"
fi

# Moving all l1b, l1c files to ODIR
mv * $ODIR/
sleep 60
rm -rf $WRK

echo "Successful Creation of MetOp files .... "
exit
set +x
```

Annexure-IV: run_l1b_plot.sh

(A wrapper to a python template: plot_hrpt_l1b_mas.py)

```
#!/bin/bash
## Plotting HRPT Level-1b AVHRR Data (Channels 1, 2, 4, 5) using python
satpy
set -x
export HME=/home/satviz/hrpt_noaa/processing_hrpt_data
export SCRP=$HME/scripts
export SRC=$HME/src
export L1B_DATA=$HME/data
export OUTD=$HME/plots/python_plots
export LOGS=/scratch/satviz/tmp/aapp/LOGS
export WRK=/scratch/${LOGNAME}/tmp/aapp/plt_l1b

source /home/satviz/Work/scripts/source/env-satpy0210

rm -rf $WRK ;mkdir -p $WRK ; cd $WRK

rm -f list.l1b  list.hrpt
rm -f hrpt_*.l1b
cp $SCRP/list.hrpt  .

## Getting the data L1B data date directories
for infile in `cat list.hrpt` ; do
ofile=`echo $infile | awk -F "/" '{print $NF}'`
odate=`echo $ofile | cut -c1-9`
oyear=`echo $ofile | cut -c1-4`
omon=`echo $ofile | cut -c6-7`
oday=`echo $ofile | cut -c8-9`
otime=`echo $ofile | cut -c11-14`

  sat=`echo $ofile | awk -F "." '{print $NF}'`

if [[ $sat == "metop-1" ]] ; then
bsat=M01
elif [[ $sat == "metop-2" ]] ; then
bsat=M02
elif [[ $sat == "noaa-15" ]] ; then
bsat=noaa15
elif [[ $sat == "noaa-18" ]] ; then
bsat=noaa18
elif [[ $sat == "noaa-19" ]] ; then
bsat=noaa19
elif [[ $sat == "noaa-20" ]] ; then
```

```
bsat=noaa20
else
bsat=$sat
fi
  echo
$L1B_DATA/$odate/hrpt_${bsat}_${oyear}${omon}${oday}_${otime}_*.l1b >>
list.l1b
done #infile
usleep 30

## Created list files with l1b datasets

while read -r file ; do

ofile=`echo $file | awk -F "/" '{print $NF}'`
asat=`echo $ofile | awk -F"_" '{print $2}' | tr 'a-z' 'A-Z'`
ymd=`echo $ofile | awk -F '_' '{print $3}'`
   year=`echo $ymd | cut -c1-4`
   mon=`echo $ymd | cut -c5-6`
   day=`echo $ymd | cut -c7-8`

   export FIGS_DIR=$OUTD/${year}${mon}${day}/AVHRR

   if [[ -f "$file" ]] ; then

[ ! -d $FIGS_DIR ] &&mkdir -p $FIGS_DIR

   ln -sf $file   .

      inf=hrpt_l1b_plot_mas.py
      sed -e 's|YEAR_EDIT|'$year'|g ; s|MON_EDIT|'$mon'|g ;
s|DAY_EDIT|'$day'|g; s|FILE_NAME|'$ofile'|g ; s|FIGS_DIR|'$FIGS_DIR'|g'
$SRC/$inf >plt_${ofile}.py
      /home/satviz/anaconda3new/envs/satpyenv0210/bin/python
plt_${ofile}.py>& $LOGS/log.plt_${ofile} &
usleep 5
done  #ch

usleep 300
    sleep 180

  fi # file check
done < list.l1b # file
set +x
exit
```

Annexure-V: run_l1c_plot.sh

```
#!/bin/bash
## Plotting HRPT Level-1c AMSU-A (Channels 1-15), MHS Data (Channels 1-
5) using python
set -x
export HME=/home/satviz/hrpt_noaa/processing_hrpt_data
export SCRP=$HME/scripts
export SRC=$HME/src
export L1C_DATA=$HME/data
export OUTD=$HME/plots/python_plots
export LOGS=/scratch/satviz/tmp/aapp/LOGS
export WRK=/scratch/${LOGNAME}/tmp/aapp/plt_l1c

source /home/satviz/Work/scripts/source/env-satpy0210

rm -rf $WRK ;mkdir -p $WRK ; cd $WRK

rm -f list.l1c  list.hrpt
rm -f amsual1c_*.l1c.h5  mhsl1c_*.l1c.h5
cp $SCRP/list.hrpt  .

## Getting the data L1B data date directories
for infile in `cat list.hrpt` ; do
ofile=`echo $infile | awk -F "/" '{print $NF}'`
odate=`echo $ofile | cut -c1-9`
oyear=`echo $ofile | cut -c1-4`
omon=`echo $ofile | cut -c6-7`
oday=`echo $ofile | cut -c8-9`
otime=`echo $ofile | cut -c11-14`

  sat=`echo $ofile | awk -F "." '{print $NF}'`

if [[ $sat == "metop-1" ]] ; then
bsat=M01
elif [[ $sat == "metop-2" ]] ; then
bsat=M02
elif [[ $sat == "noaa-15" ]] ; then
bsat=noaa15
elif [[ $sat == "noaa-18" ]] ; then
bsat=noaa18
elif [[ $sat == "noaa-19" ]] ; then
bsat=noaa19
elif [[ $sat == "noaa-20" ]] ; then
bsat=noaa20
```

```
else
bsat=$sat
fi
  echo
$L1C_DATA/$odate/amsual1c_${bsat}_${oyear}${omon}${oday}_${otime}_*.l1c.
h5 >> list.l1c
  echo
$L1C_DATA/$odate/mhsl1c_${bsat}_${oyear}${omon}${oday}_${otime}_*.l1c.h5
>> list.l1c
done #infile
usleep 30

## Created list files with l1b datasets

while read -r file ; do

ofile=`echo $file | awk -F "/" '{print $NF}'`
asat=`echo $ofile | awk -F"_" '{print $2}' | tr 'a-z' 'A-Z'`
ftype=`echo $ofile | awk -F '_' '{print $1}'`
ymd=`echo $ofile | awk -F '_' '{print $3}'`
   year=`echo $ymd | cut -c1-4`
   mon=`echo $ymd | cut -c5-6`
   day=`echo $ymd | cut -c7-8`

   export FIGS_DIR=$OUTD/${year}${mon}${day}/AMSU

   if [[ -f "$file" ]] ; then

[ ! -d $FIGS_DIR ] &&mkdir -p $FIGS_DIR

   ln -sf $file   .

   if [[ "$ftype" == "amsual1c" ]] ; then
nch=15
elif [[ "$ftype" == "hrpt" ]] ; then
nch=5
elif [[ "$ftype" == "mhsl1c" ]] ; then
nch=5
   fi

    for ch in $(eval echo "{1..$nch}") ; do
      /home/satviz/anaconda3new/envs/satpyenv0210/bin/python
$SRC/aapp1c_plot.py  -i $ofile -c $ch -a -H >&
$LOGS/log.plt_${ofile}_ch${ch}
usleep 5
```

```
    mv ${ofile}_${ch}.png  $FIGS_DIR/
done  #ch
usleep 300
    unlink $file

  fi # file check
done < list.l1c # file


set +x
```

Annexure-VI: plot_hrpt_l1b_mas.py

```
#!/usr/bin/env python
'''
What does this script do?
This python script takes NOAA-AVHRR Level-1b data processed by AAPP
and plots it using satpy. This script is governed by a wrapper shell
script written by Dr. D. Srinivas and this python script is written
by Dr. M. Sateesh. This script has been edited to PEP8 standards by
MNRS.

'''
# Start importing
import os, sys
os.environ['PPP_CONFIG_DIR'] = '/home/satviz/.local/satpy/etc/'
from satpy import Scene
from glob import glob
from satpy.utils import debug_on
debug_on()
import numpy as np
import aggdraw

# Start logic
daate = 'YEAR_EDIT.MON_EDITDAY_EDIT'
date = 'YEAR_EDITMON_EDITDAY_EDIT'

# Set up base folders
font_dir = '/home/satviz/Work/fonts/'
shape_dir = '/home/satviz/Work/GSHHS/India/'
base_dir = 'FIGS_DIR/'
web_dir = '/home/satviz/Work/web/' + date + '/'
logo_dir = '/home/satviz/Work/logo/'

# Start reading files
files = glob("./" + "FILE_NAME")
if(len(files)>0):
 for filename in files:
   scn = Scene(filenames=[filename], reader='avhrr_l1b_aapp')
   channels = ['Channel1','Channel2','Channel4','Channel5']
   scn.load(channels)
   indImg = scn.resample('India_SC30')
   ## For platform name ###
   platform_name = scn['channel1'].attrs['platform_name']
   start_time= scn['channel1'].attrs['start_time']
   end_time= scn['channel1'].attrs['end_time']
```

```
   for channel in channels:
       indImg[channel].attrs['platform_name'] = platform_name
       indImg[channel].attrs['start_time'] = start_time
       print(channel)
       indImg.save_datasets(base_dir = base_dir,
filename='ind_{platform_name}_{sensor}_{name}_{start_time:%Y%m%d_%H%M%S}
.png',compute=True,writer='simple_image',datasets=[channel],
                            overlay = {'coast_dir': shape_dir,
                                       'resolution':'i','level_coast1':
1,'level_borders':1,'level_coast2': 7,'width1':1,

'width2':0.2,'width':0.8,'color':'white',
                                       'grid': {'major_lonlat': (10,
10),'minor_lonlat':(2,2),

'fill_opacity':255,'minor_is_tick':True,'minor_width':0.3,
                                       'write_text': True,
                                       'outline': (224, 224, 224),
                                       'width':
0.8,'outline':'lightblue',
                                       'font': aggdraw.Font('white',
font_dir + 'DejaVuSerif.ttf', opacity=255, size=30)}},
                            decorate={'decorate': [
                                {'logo': {'logo_path': logo_dir +
'NCMRWF.png', 'height': 122, 'bg': 'white',
                                          'bg_opacity': 255, 'align':
{'top_bottom': 'top', 'left_right': 'left'}}},
                                {'text': {'txt': '\n ' + platform_name +
': avhrr-3\n' + str(start_time),
                                         'align': {'top_bottom': 'top',
'left_right': 'left'},
                                          'font': font_dir +
'DejaVuSerif.ttf',
                                          'font_size': 32, 'height': 30,
                                          'bg': 'yellow', 'bg_opacity':
170, 'line': 'blue'}},
                                {'logo': {'logo_path': logo_dir +
'imdlogo.jpg', 'height': 120, 'bg': 'white',
                                           'bg_opacity': 255, 'align':
{'top_bottom': 'top', 'left_right': 'right'}}}
                               ]})
else:
  print("Skipped")
# EOF
```

Annexure–VII: aapp1c_quicklook.py

```
# Python script to plot level 1c/1d hdf5 files of AMSU and MHS channels
Brightness temperatures.
#!/usr/bin/env python

"""$Id: aapp1c_quicklook.py 626 2019-03-12 10:37:57Z frna $

Quicklook display of AAPP level 1c/1d hdf5 data from sounders

Input: AAPP .h5 file

Output: To screen or png file or both

Projection: Plate Carree (i.e. equirectangular, true at equator)

Requirements: python2.7 or 3.x, with numpy, matplotlib, h5py, cartopy.
If you don't have cartopy, comment out the lines that refer to ccrs; it
will still display the BT data.

For usage instructions, type

aapp1c_quicklook.py -h

Works with AAPP-generated l1c.h5 or l1d.h5 files (see convert_to_hdf5),
for instruments such as:
  AMSU, MHS, HIRS, ATMS, MWHS2, MWTS2, IRAS, CrIS, IASI

BT will be displayed if available, otherwise radiance (CrIS) or scaled
radiance (IASI)

COPYRIGHT
   This software was developed within the context of the EUMETSAT
Satellite
   Application Facility on Numerical Weather Prediction (NWP SAF), under
the
   Cooperation Agreement dated 7 December 2016, between EUMETSAT and the
   Met Office, UK, by one or more partners within the NWP SAF. The
partners
   in the NWP SAF are the Met Office, ECMWF, DWD and MeteoFrance.

   Copyright 2018, EUMETSAT, All Rights Reserved.

History:
Version    Date     Comment
```

```
1.0    11/03/2019  Initial. NCA

Edited by: Sateesh, Srinivas Desamsetti; NCMRWF.
03/June/2020
"""

import numpy as np
import numpy.ma as ma
import h5py as h5
import matplotlib.pyplot as plt
import sys
import cartopy.crs as ccrs
from argparse import ArgumentParser
from mpl_toolkits.basemap import Basemap

parser = ArgumentParser()
parser.add_argument("-i", "--in", required=True, dest="inputfile",
help="input h5 file")
parser.add_argument("-c", "--channel", dest="channel", default="1",
help="Channel to display")
parser.add_argument("-r", "--range", dest="latlonrange", default='-180
180 -90 90', help="geographical limit: 'x0 x1 y0 y1', default '-180 180
-90 90'")
parser.add_argument("-s", "--symsize", dest="symsize", default=4,
help="symbol size, default 4")
parser.add_argument("-nd", "--no_display", dest="display_image",
action="store_false", default=True, help="don't display image on
screen")
parser.add_argument("-ns", "--no_save", dest="save_image",
action="store_false", default=True, help="don't save image")
parser.add_argument("-a", "--auto", dest="auto_region",
action="store_true", default=False, help="auto range for lat/lon")
parser.add_argument("-H", "--high_resolution", dest="high_res",
action="store_true", default=False, help="high resolution coastline")

args = parser.parse_args()

# Read arrays from h5 file

fid  =  h5.File(args.inputfile, 'r')
group  =  fid['/']
LATITUDE = 1.0E-4*np.ma.array(group['Geolocation/Latitude'])
LONGITUDE =  1.0E-4*np.ma.array(group['Geolocation/Longitude'])
btok = "/Data/btemps" in fid
radok = "/Data/radiance" in fid
```

```
scalradok = "/Data/scalrad" in fid
if btok:
    print ("Reading Data/btemps")
    BT = 1.0E-2*np.ma.array(group['Data/btemps'])
elifradok:
    print ("Reading Data/radiance")
    BT =  1.0E-4*np.ma.array(group['Data/radiance'])    #Convert to
mW/m^2/sr/cm^-1
elifscalradok:
    print ("Reading Data/scalrad")
    BT =  1.0*np.ma.array(group['Data/scalrad'])     #no scaling for
scalrad
fid.close()

BT = ma.masked_where(BT <= 10.0, BT)

# Extract the required channel

if (len(BT.shape) == 2):
image_array = BT[:,int(args.channel)-1]
elif (len(BT.shape) == 3):
image_array = BT[:,:,int(args.channel)-1]
elif (len(BT.shape) == 4):
image_array = BT[:,:,:,int(args.channel)-1]

# Convert the lat/lon bounds to an array

if args.auto_region:
latlonrange = np.fromstring("0 0 0 0", dtype=float, sep=' ')
latlonrange[0] = np.amin(LONGITUDE)
latlonrange[1] = np.amax(LONGITUDE)
latlonrange[2] = np.amin(LATITUDE)
latlonrange[3] = np.amax(LATITUDE)
else:
latlonrange = np.fromstring(args.latlonrange, dtype=float, sep=' ')

m = Basemap(projection='cyl', resolution='l',llcrnrlat=latlonrange[2],
urcrnrlat = latlonrange[3],llcrnrlon= latlonrange[0], urcrnrlon =
latlonrange[1])
lon = LONGITUDE
lat = LATITUDE
data = image_array
###########
mask = ((LONGITUDE >= latlonrange[0]) & (LONGITUDE <= latlonrange[1]) &
```

```
        (LATITUDE >= latlonrange[2]) & (LATITUDE <= latlonrange[3]) &
(image_array>= 0))
LATITUDE=LATITUDE[mask]
LONGITUDE=LONGITUDE[mask]
image_array = image_array[mask]

# Change the default 8x6 inch plot
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 18
fig_size[1] = 9
plt.rcParams["figure.figsize"] = fig_size

# Plot the map
ax = plt.axes(projection=ccrs.PlateCarree())
ax.set_extent(latlonrange,crs=ccrs.PlateCarree())
if args.high_res:
   ax.coastlines('50m')
else:
   ax.coastlines()
   gl = ax.gridlines(crs=ccrs.PlateCarree(), linewidth=1,
color='gray', linestyle='--', draw_labels=True)
   gl.xlabels_top = None

# Plot the data
cmap="jet"       #alternative: "RdBu_r"
#plt.scatter(LONGITUDE,LATITUDE,s=int(args.symsize),c=image_array,
marker="o", lw=0, cmap=cmap)
#plt.contourf(LONGITUDE,LATITUDE,image_array)
m.pcolormesh(lon, lat, data, latlon=True, cmap='jet')
plt.title(args.inputfile)

# Draw the colour bar
cbar = plt.colorbar(orientation='vertical')
if btok:
cbar.ax.set_ylabel('Channel '+args.channel+' BT (K)')
elifradok:
cbar.ax.set_ylabel('Channel '+args.channel+' radiance
(mW/m$^2$/sr/cm$^{-1}$)')
elifscalradok:
cbar.ax.set_ylabel('Channel '+args.channel+' scaled radiance')
# Save or display image
if args.save_image:
outfile=args.inputfile+"_"+args.channel+".png"
plt.savefig(outfile)
print(("created "+outfile))
```

Annexure–VIII: run_plot_time_delays.sh

```
#!/bin/bash
# Data monitoring master submission script
set -x
export DATAD=/home/hrpt_incois
export
OUTDIR=/home/satviz/hrpt_noaa/processing_hrpt_data/plots/data_recep_dela
y
export SRC=/home/satviz/hrpt_noaa/processing_hrpt_data/src
export WRK=/scratch/${LOGNAME}/tmp/aapp/delay_plot

module load gnu/rpackages/3.4.4

mkdir -p $WRK ; cd $WRK
cp $SRC/plt_delay.r
cp $SRC/plt_delay_mon.r

#cdate=20200518
cdate=`date +%Y%m%d`
dd=2   #2-days ago

rdate=`date -d "$cdate - ${dd} days" +%Y%m%d`
sdate=`date -d "$cdate - ${dd} days" +%Y-%m-%d`
year=`date -d "$cdate - ${dd} days" +%Y`
mon=`date -d "$cdate - ${dd} days" +%m`
day=`date -d "$cdate - ${dd} days" +%d`
outdt=`date -d "$cdate - ${dd} days" +%Y-%m`

echo "Processing Date - $rdate"

ls -l $DATAD/${year}.${mon}${day}.* > tmp.txt
/home/apps/SiteSoftwares/gnu/R.3.4.3/bin/Rscriptplt_delay.r  $sdate
export OUTD=$OUTDIR/$outdt ;mkdir -p $OUTD
mv out.png  $OUTD/hrpt_time_delay_${rdate}.png

# Generate monthly time delay figures
if [ $day -eq 3 ]; then
dd=8   #2-days ago
rdate=`date -d "$cdate - ${dd} days" +%Y%b`
sdate=`date -d "$cdate - ${dd} days" +%b-%Y`
year=`date -d "$cdate - ${dd} days" +%Y`
mon=`date -d "$cdate - ${dd} days" +%m`
outdt=`date -d "$cdate - ${dd} days" +%Y-%m`
```

```
ls -l $DATAD/${year}.${mon}??.* > tmp.txt
/home/apps/SiteSoftwares/gnu/R.3.4.3/bin/Rscriptplt_delay_mon.r  $sdate
export OUTD=$OUTDIR/$outdt ;mkdir -p $OUTD
mv out.png  $OUTD/hrpt_time_delay_${rdate}.png
fi
set +x
```

Annexure–IX: plt_delay.r

```
# Data monitoring script – will run with R software
library(stringr)
args=commandArgs(trailingOnly=TRUE)
system("export TZ=UTC")

ss <- read.csv(file="./tmp.txt",header=F)
ss$sdate<- substr(ss$V1,43,48) # system date/reception date
ss$stime<- substr(ss$V1,50,54)
ss$rdate<- substr(ss$V1,74,82) # real date/file date
ss$rtime<- substr(ss$V1,84,89)
ss$file<- substr(ss$V1,56,115)
ss$file2  <- substr(ss$V1,84,89)

ss$file1 <- format(as.POSIXct(ss$file2, "%H%M%S",tz="UTC"),"%H:%M:%S")
ss$file<- paste(ss$file1,str_sub(ss$file, str_length(ss$file)-6, -
1),sep="-")

ss$rdatetime<- as.POSIXct(paste(ss$rdate,ss$rtime,sep=""),
"%Y.%m%d%H%M%S",tz="UTC")
ss$sdatetime<- as.POSIXct(paste(ss$sdate,ss$stime,sep=" "), "%b %d
%H:%M",tz="Asia/Calcutta")

ss$diff<- difftime(ss$sdatetime, ss$rdatetime,units="hours")

plotname = "./out.png"

png(plotname,width = 5*300, height = 6*300, res=150)
par(mar=c(4,10,2,2))
barplot(as.numeric(ss$diff),names.arg=ss$file,horiz=TRUE,density=10,xlab
="Time(hours)",font.axis=2,main=paste("HRPT Data Delay at NCMRWF : ",
args[1],sep=""),las=1,inside = TRUE,cex.axis=1.5, cex.names=1.2, cex.lab
= 1.8, cex.main=1.8)

## Add vertical lines at 30 min and 1 hour
abline (v=c(0.5,1), col=c("blue","red"), lty=c(1,2), lwd=c(1,2))
dev.off()
quit()
```