Journal of Computational Science 81 (2024) 102363

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Python Indian Weather Radar Toolkit (pyiwr): An open-source Python library for processing, analyzing and visualizing weather radar data

Nitig Singh [a], Vaibhav Tyagi [a], Saurabh Das [a,*], Udaya Kumar Sahoo [b], Shyam Sundar Kundu [c]

[a] Indian Institute of Technology Indore, Madhya Pradesh, India
[b] Indian Institute of Tropical Meteorology, (IITM), Sehore, Madhya Pradesh, India
[c] North Eastern Space Applications Centre, (NESAC), Umiam, Meghalaya, India

ARTICLE INFO

Keywords:
Python toolkit
DWR
CAPPI
QPE
Data processing
Time series analysis

ABSTRACT

The Python Indian Weather Radar Toolkit, abbreviated as "pyiwr", is an open-source Python library tailored for the purpose of handling data from the Indian Doppler Weather Radar (DWR). This paper provides a comprehensive overview of the pyiwr, which serves as a toolkit to read, analyze, process, and visualize weather radar data. Apart from this, the toolkit offers a range of robust functions implementing various algorithms covering several aspects of the radar data processing and quality control that facilitate the manipulation and analysis of weather radar data. To demonstrate the practical applicability of pyiwr, various case studies are presented, focusing on processing raw reflectivity data (clutter correction), Quantitative Precipitation Estimation (QPE) using Z-R relationship and time-series analysis of reflectivity and rain intensity, both spatially as well as at a specific location, during various meteorological events. This module enhances the accessibility and compatibility of radar data, enabling researchers, weather forecasters, and hydrologists to efficiently work with DWR data (particularly Indian DWR) that fosters advancements in weather radar research and applications. The open availability of pyiwr's source code on GitHub ensures that researchers and practitioners can not only access the toolkit but also contribute to its ongoing development.

1. Introduction

Weather radar systems are instrumental in modern meteorology, providing crucial data for monitoring, predicting, and comprehending weather patterns. The Doppler Weather Radar (DWR) holds a crucial role in enhancing nowcasting applications, allowing for more accurate and timely short-term weather predictions. Furthermore, the assimilation of this DWR data into mesoscale Numerical Weather Prediction (NWP) models serves as a vital role in improving the precision and reliability of these models for localized weather forecasting [2]. The continuous advancements in radar technology have resulted in the accumulation of wealth of vast valuable radar data, yet extracting meaningful insights requires considerable effort [18,22]. Processing, visuaizing, and analyzing weather radar data encompass a broad spectrum of computer disciplines and fields of study. With the vast array of information available, the development of adaptive and scalable software solutions becomes paramount, thus necessitating the development of efficient tools for data processing and analysis. Such software must cater to the diverse needs of users, spanning from basic data visualization to the construction of intricate processing workflows. Ideally, these tools should strike a balance, offering simplicity for routine tasks while empowering experienced users with advanced functionalities. They offer ease of installation and usability for routine tasks, while also providing advanced users with access to powerful features. However, the complexities associated with radar data processing pose several challenges in its effective and widespread use.

Radar data processing encounters a primary challenge due to the diverse complex file formats of radar data, and their compatibility with the existing radar data processing toolboxes. The processing, calibration, and analysis of weather radar data involve various computational disciplines and techniques, often requiring substantial prior knowledge and specialized skills, making it difficult to implement various algorithms effectively thus, constraining its usage. While the radar instrument handles some signal processing and quality control tasks, additional software is often required for more comprehensive processing and analysis [8].

The advent of radar polarimetry enhances radar capabilities by providing detailed precipitation target characteristics like shape, size,

* Corresponding author.
E-mail addresses: das.saurabh01@gmail.com, saurabh.das@iiti.ac.in (S. Das).

https://doi.org/10.1016/j.jocs.2024.102363
Received 27 September 2023; Received in revised form 18 April 2024; Accepted 11 June 2024
Available online 13 June 2024

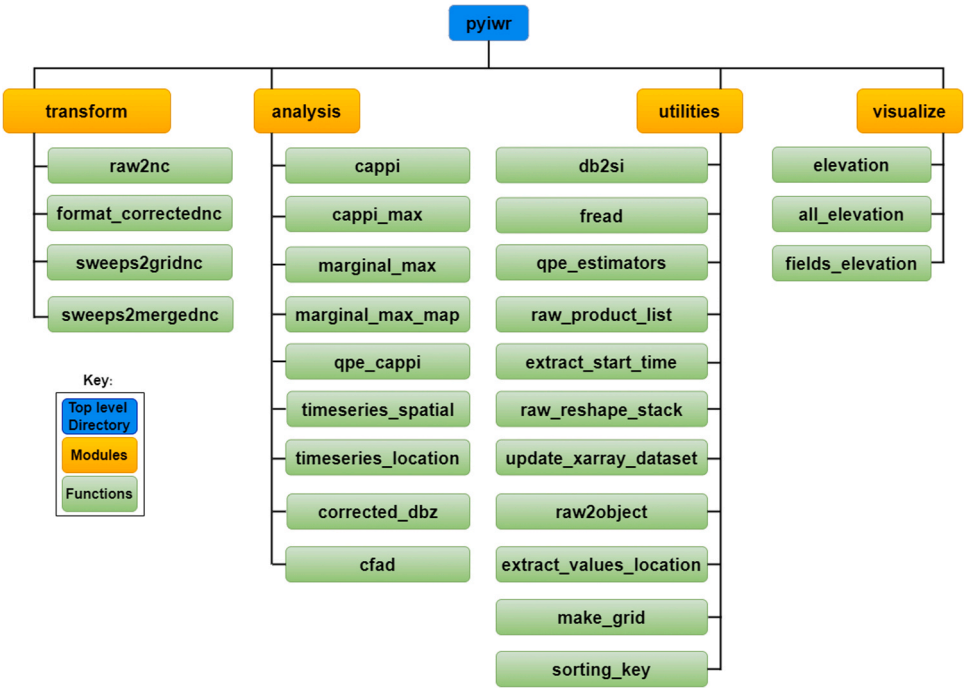

Fig. 1. Architecture of the pyiwr.

and phase. Consequently, researchers focus on polarimetric algorithms to enhance radar rainfall estimates. Methods like the self-consistent approach calibrate radar reflectivity using polarimetric relationships. Weather radar does not directly measure rainfall rates; rather, they are inferred from the radar reflectivity using the empirical relationship for rainfall estimation [3,12,19,20]. Recent studies demonstrate the efficacy of polarimetric rain rate estimators, providing more accurate rainfall estimates under diverse radar frequencies and geographical conditions. Other methodologies identify non-meteorological echoes and improve data quality [21]. Various factors can affect the accuracy of weather radar data quality. These include radar miscalibration, obstruction of the radar beam, issues related to beam propagation such as ground clutter [1,7].

Over the years, the radar community has dedicated extensive efforts to crafting algorithms aimed at mitigating the influence of these error sources and enhancing rainfall estimation derived from radar data. Despite significant advancements in this area, residual errors persist in radar rainfall estimates, posing challenges for downstream applications such as flood forecasting. These errors can propagate through various models that rely on radar rainfall products, underscoring the ongoing need for improvement in radar data processing and analysis methodologies. To address these challenges and foster advancements in radar data handling and analysis, several open-source Python modules have been developed to facilitate radar data processing and analysis. Py-ART, an open-source Python library known as the Python ARM Radar Toolkit, has gained widespread adoption for working with weather radar data [8]. Py-ART provides essential functionalities for reading, processing, and visualizing radar data in various formats, contributing significantly to radar data analysis. Likewise, wradlib, another widely used Python library, focusing on the processing of weather radar data, emphasizing on geospatial analysis. It offers capabilities like noise and artifact elimination, along with range folding and beamwidth correction functions [9]. Similar in line, Towerpy, a python library, specialized in handling polarimetric weather radar data, offers various robust algorithms for both radar data quality control and radar quantitative precipitation estimation (QPE) through a dedicated processing chain [22]. Pyrad, another library which is a real-time framework for reading, processing, and visualizing polar and cartesian radar data from multiple radars, operates both in offline and in real time for data processing [30]. A recent noteworthy development for effectively managing India Meteorological Department (IMD) radar data is the introduction of PyScanCf. This specialized library is designed to generate cfradial data from IMD radars, accommodating both individual scan files and gridded radar data. It also seamlessly integrates with Py-ART, further enhancing its compatibility and utility [38].

The open-source libraries mentioned above have brought together a growing community of radar researchers from around the world. They've made it easier for researchers to work with radar data, allowing them to create new radar algorithms that can be easily reproduced by others. These algorithms are also making their way into the software used by operational weather radar networks. Although these existing radar toolkits are valuable, they may not fully cater to the unique characteristics and challenges when it comes to Indian DWR data. Accessing and processing radar data from the Indian weather radar network, particularly from Indian Space Research Organization (ISRO), remains a formidable challenge for open-source radar researchers. The current raw data format employed by ISRO is incompatible with existing open-source libraries, rendering it unreadable and un-processable. Furthermore, the lack of specific algorithms designed to work with open source radar's data makes the problem worse, which have some critical issues like missing data sweeps and inaccurate datetime information. Furthermore, the installation process for existing libraries presents hurdles due to extensive dependencies, high Central Processing Unit (CPU) usage, and time constraints. Additionally, the manipulation of volumetric data essential for various pseudo-surveillance strategy analyses remains largely unaddressed by many libraries [14]. Another obstacle confronting the Indian radar community is the incongruity of radar file formats, hindering direct utilization of existing libraries. Additionally, the requisite proficiency in programming languages poses a significant barrier for scientists and researchers seeking to navigate radar data challenges effectively.

In response to the numerous challenges associated with Indian Weather Radar data, the Python Indian Weather Radar Toolkit (pyiwr) has been developed as an open-source Python module by researchers at the SIGMA Research Lab, Indian Institute of Technology Indore. This toolkit offers to process polarimetric radar data, and is specifically

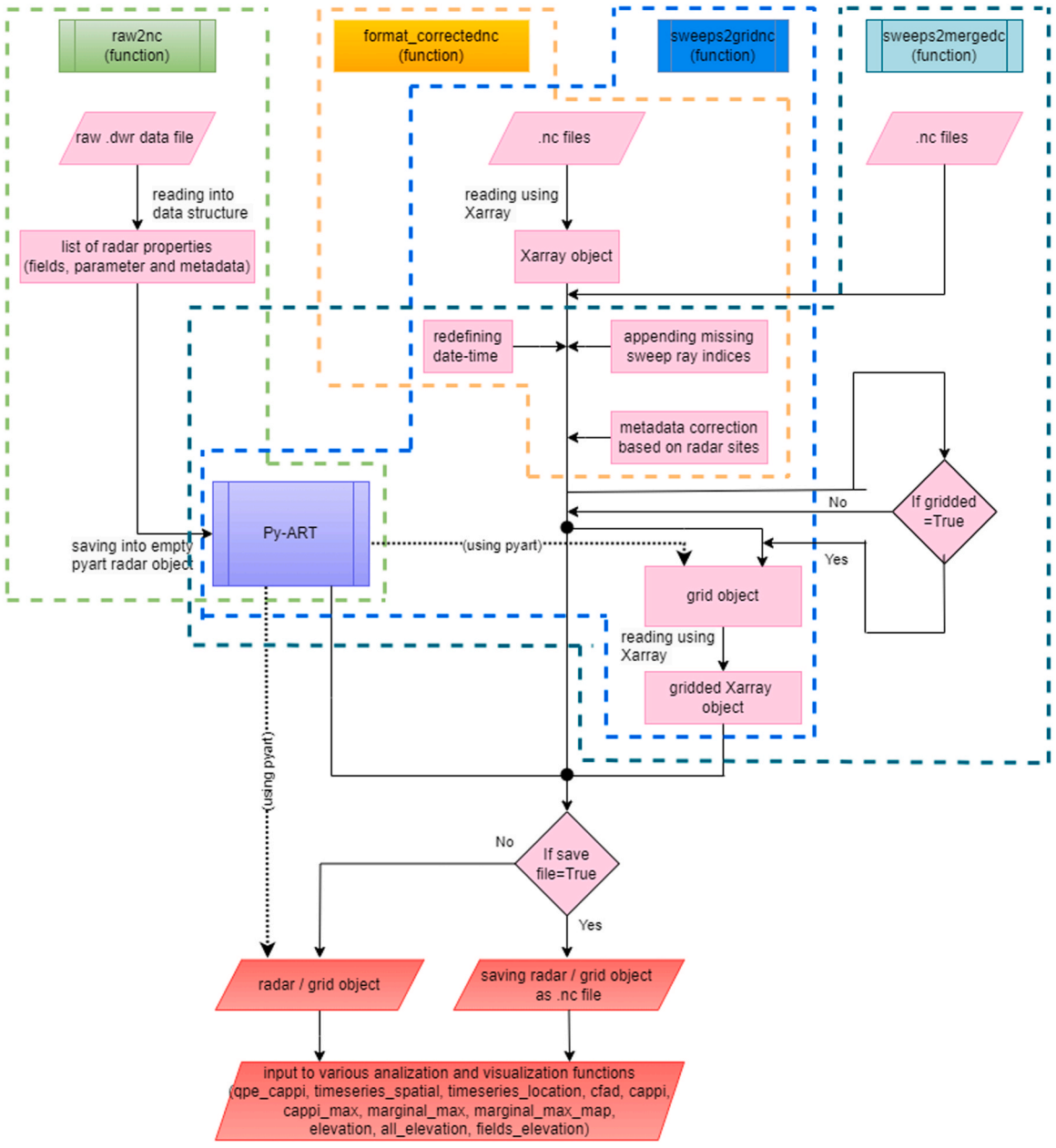

Fig. 2. Workflow of transform module in pyiwr.

designed to meet the needs and issues faced by the Indian radar research community. This tool incorporates standard procedures for processing and visualizing polarimetric weather radar data, making it easier for radar users to work with raw radar data, resolve format issues, convert volumetric data into gridded data, and visualize various radar products. Additionally, it provides algorithms for analyzing radar products vertically, spatially, and over time. Users can also test different radar polarimetric algorithms to improve radar quantitative precipitation estimation (QPE), correct the reflectivity using polarimetric products and evaluate their impact. The primary objective of pyiwr is to streamline the data parsing process of raw DWR files, and seamlessly restructuring and resolving issues associated with NetCDF DWR files, enhancing data accessibility and usability for researchers, weather forecasters, and hydrologists. The toolkit offers one-liner commands for each function, ensuring flexibility and ease of use. With its straightforward installation process and user-friendly interface, this library has the potential to become a comprehensive solution for the Indian radar community.

The paper is organized into distinct sections. Section 2 provides an overview of the radar systems and data products. In Section 3, describes the architecture of the pyiwr followed by the detailed exploration of the methodology and implementation aspects explaining the usage of the various functions, as described in the Section 4. Section 5 illustrates potential use cases of pyiwr using various case study that exemplifies the practical application of the package. Subsequently, the discussion and outlook in Section 6. For comprehensive installation instructions and documentation, please refer to the appendix.

2. Radar systems and data

Doppler Weather Radar (DWR) is an essential tool for weather monitoring and prediction on a continuous basis. DWRs have been used for atmospheric research and operational purposes for many decades and have revolutionized weather forecasting and nowcasting. Doppler radars help in observing the extent and intensity of rainfall and cloud formations, and monitoring cyclones, thunderstorms and lightning in

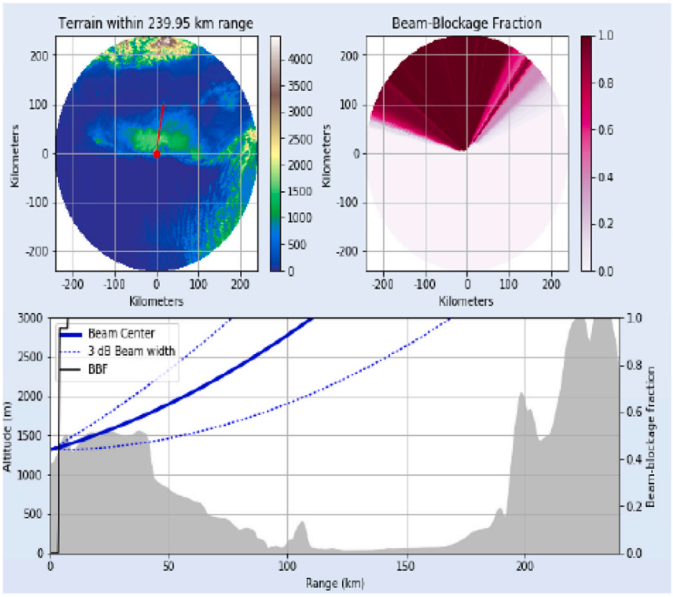

Fig. 3. Beam blockage calculation for Cherrapunji radar at 10 degrees azimuth. [Image from [17]].

real time. Currently, IMD hosts a wide network of 37 DWRs (working in X, C and S bands) installed across India. Radar systems across various frequency bands, such as X, S, and C bands, provide a diverse range of meteorological data products essential for weather analysis and forecasting. These include [3,6]:

- Reflectivity (DBZ): Reflectivity data represent the intensity of returned radar signals, providing information about the concentration and size of hydrometeors in the atmosphere.
- Radial Velocity (VEL): Radial velocity data indicate the speed and direction of motion of precipitation particles relative to the radar, aiding in the detection of wind patterns and atmospheric circulation.
- Spectral Width (WIDTH): Spectral width measurements characterize the variability in the radial velocities of hydrometeors, offering insights into turbulence, particle size distribution, and atmospheric instability.
- Differential Reflectivity (ZDR): Differential reflectivity is a polarimetric parameter that compares the returned power of horizontally and vertically polarized radar signals, assisting in the identification of hydrometeor types and particle shape information.
- Differential Phase (PHIDP): Differential phase represents the difference in phase shifts between horizontal and vertical radar signals, providing valuable information about particle size distribution, liquid water content, and precipitation processes.
- Correlation Coefficient (RHOHV): Correlation coefficient data indicate the similarity between received polarized radar signals, offering insights into the uniformity and orientation of hydrometeors, as well as distinguishing between meteorological and non-meteorological targets.
- Specific Differential Phase (KDP): Specific differential phase represents the rate of change of differential phase with respect to range, providing information about rainfall intensity, precipitation type, and liquid water content.

The present study utilizes open-source DWRs data. The developed package "pyiwr" has been extensively tested for three open DWR datasets including TERLS, C-band radar stationed at Thumba Equatorial Rocket Launching Station (TERLS) in Thiruvananthapuram, CHERRAPUNJI, S-band radar located in the renowned rainfall-prone region of Cherrapunji and SHAR, S-band radar situated at Satish Dhawan Space Centre-Sriharikota Range (SHAR) in Sriharikota. Apart from the open DWR datasets, the package is tested with other radar datasets available across the country.

3. Architecture of pyiwr

"pyiwr" is specifically designed to process, correct, analyze and visualize Indian DWR data. The library functions are well designed to support and make the reading and visualization of radar formats very smooth. Thus, more focused can be given to the understanding and deriving science from the data, as a considerable amount of time is spent in understanding reading and visualization of the radar data and to make it compatible with existing toolkits.

Python [23], a high-level interpreted language recognized for its expressive and understandable syntax, is used in the development of pyiwr. Python is simple to learn and well-documented, with a big built-in standard library and a wide selection of third-party packages for scientific computing. "pyiwr" becomes platform-independent by building on top of Python, necessitating the fulfillment of package requirements on various platforms like windows and Linux. "pyiwr" enhances its usefulness by using the power of several Python packages.

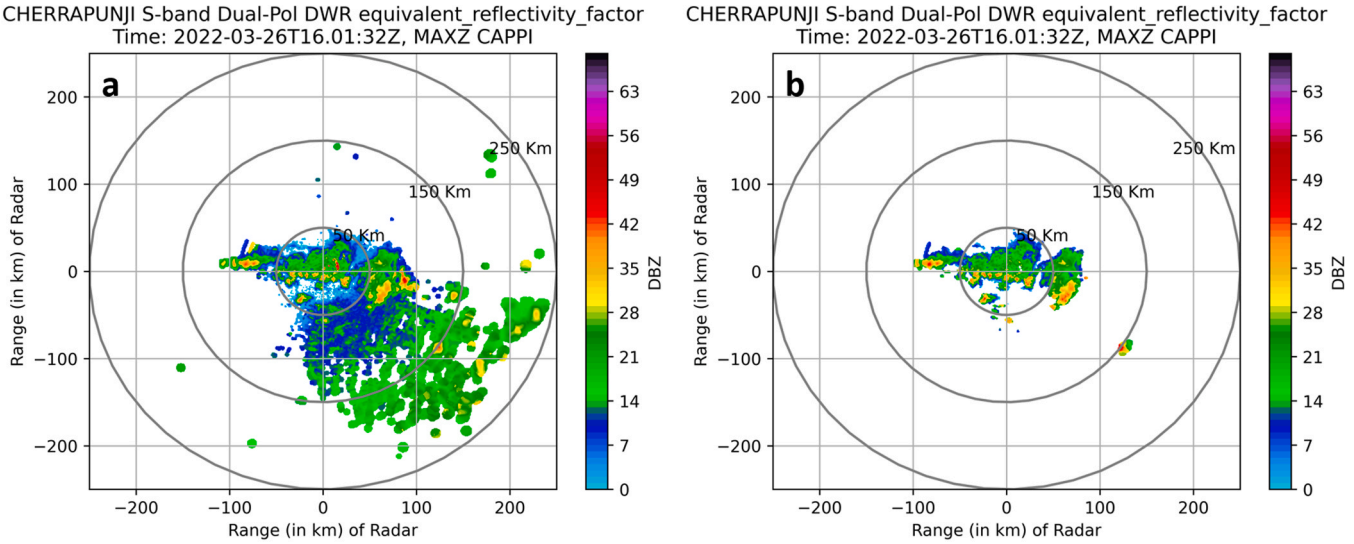

Fig. 4. Radar reflectivity (MAXZ) a.) before correction, b.) after correction for Cherrapunji DWR at March 26, 2022 16:01:32 (UTC).

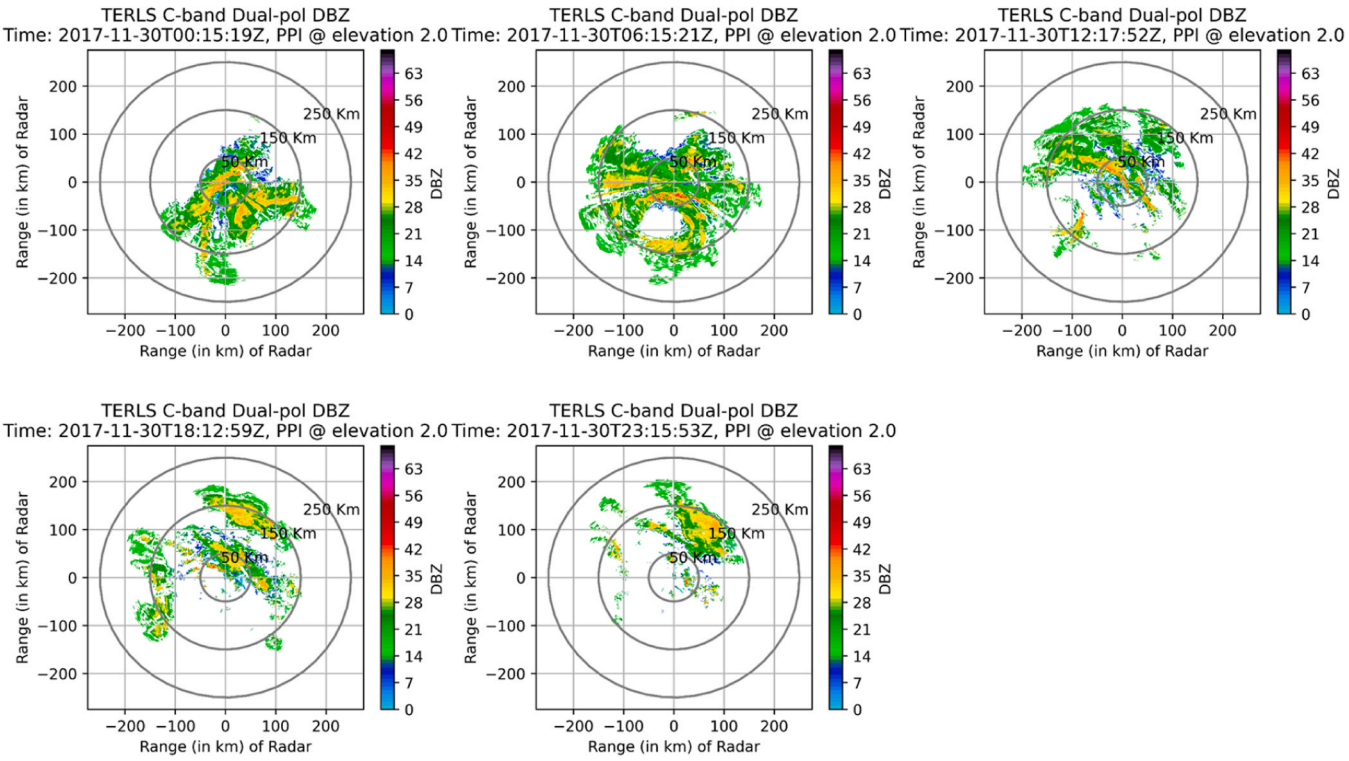

Fig. 5. 6 hourly time-series of DWR reflectivity for November 30, 2017 from 00:15:19 UTC to 23:15:53 UTC at 2° elevation.

The NumPy [16], package is used to hold multi-dimensional arrays of numerical radar data in memory effectively, allowing for high-speed operations. It makes use of matplotlib [10], a popular Python tool in the scientific community, for visualization. Pyiwr also makes use of Cartopy [13], which is based on matplotlib, to generate interactive and high-quality maps, which makes it ideal for displaying radar field plots and geographical data. Furthermore, pyiwr supports NetCDF files [29], which is one of the standard weather radar data formats around the globe. "pyiwr" makes use of the netcdf4-python [34] package to read and write these files.

As it is widely known that reading and analyzing cryptic radar format data might be difficult for users therefore pyiwr supports data formats popular in India, notably raw.dwr and NetCDF. Furthermore, based on user demand and the availability of appropriate documentation, pyiwr intends to expand its support for other formats in the future. "pyiwr" uses Xarray [25], a Python library developed for working with labeled multi-dimensional arrays (ND arrays), for Cartesian grids. Xarray not only provides powerful analytics and visualization capabilities, but it also has parallels to pandas, making it user-friendly and approachable. Xarray returns metadata in the form of dictionary objects, which provide important information about the data and may be browsed and inspected using keywords. This level of metadata adaptability enables users to tailor the data structure to the requirements and complexity of the given data source. The pyiwr architecture is designed as a user-friendly toolkit, specifically tailored to address the distinctive requirements of Indian DWR data analysis. The pyiwr is subdivided into four modules namely "transform", "visualize", "analysis" and "utilities" which serves as a vital role in streamlining the radar data processing workflow as shown in Fig. 1. Each module consists of various functions designed to perform specific tasks.

4. Methodology and Implementation

At its core, pyiwr is organized into four primary modules: "*transform.py*", "*analysis.py*", "*utilities.py*" and "*visualize.py*". The subsequent subsection will focus on the methodology in brief, providing description of the methods employed and how they were implemented.

4.1. *transform.py*

The workflow of the transform module which serves as the backbone of the library is shown in Fig. 2. It consists of four functions "*raw2nc*", "*format_correctednc*", "*sweeps2gridnc*" and "*sweeps2mergednc*" which enable the users to make the radar object or grided object assigned to a variable and if needed, save them into a NetCDF file by giving the parameter for the same.

The "*raw2nc*" function extracts the unstructured raw radar data into list data structure of NumPy arrays. This list contains radar fields, parameters, and attributes, which are then saved into an empty radar object by redefining them from scratch into the standard radar NetCDF file format using Py-ART. On the other hand, "*format_correctednc*" function takes in any DWR radar NetCDF file and restructures it. The function makes an Xarray and checks/corrects/adds any missing radar parameters and attributes like sweep start and end ray indices, metadata, and date-time corrections, if any, and write it as radar object to be used for further processing and visualization.

A similar restructuring is done in the "*sweeps2gridnc*" function before making a cartesian grid object from a NetCDF file using Py-ART, which is then saved as a gridded Xarray object. The volumetric plan position indicator (PPI) data at various elevations is converted into georeferenced gridded constant altitude PPI data. The PPI data in range, azimuth and elevation (r, θ_a, θ_e) is converted into cartesian (x, y and z) and geographic (latitude, longitude and altitude). The computation of Cartesian coordinates follows Eqs. 1, 2, and 3, assuming a standard atmosphere (using the 4/3 Earth's radius model) [35].

$$z = \sqrt{r^2 + R^2 + 2rR\sin\theta_e} - R \tag{1}$$

$$s = R\arcsin\left(\frac{r\cos\theta_e}{R+z}\right) \tag{2}$$

$$x = s\sin\theta_a \;\; ; \;\; y = s\cos\theta_a \tag{3}$$

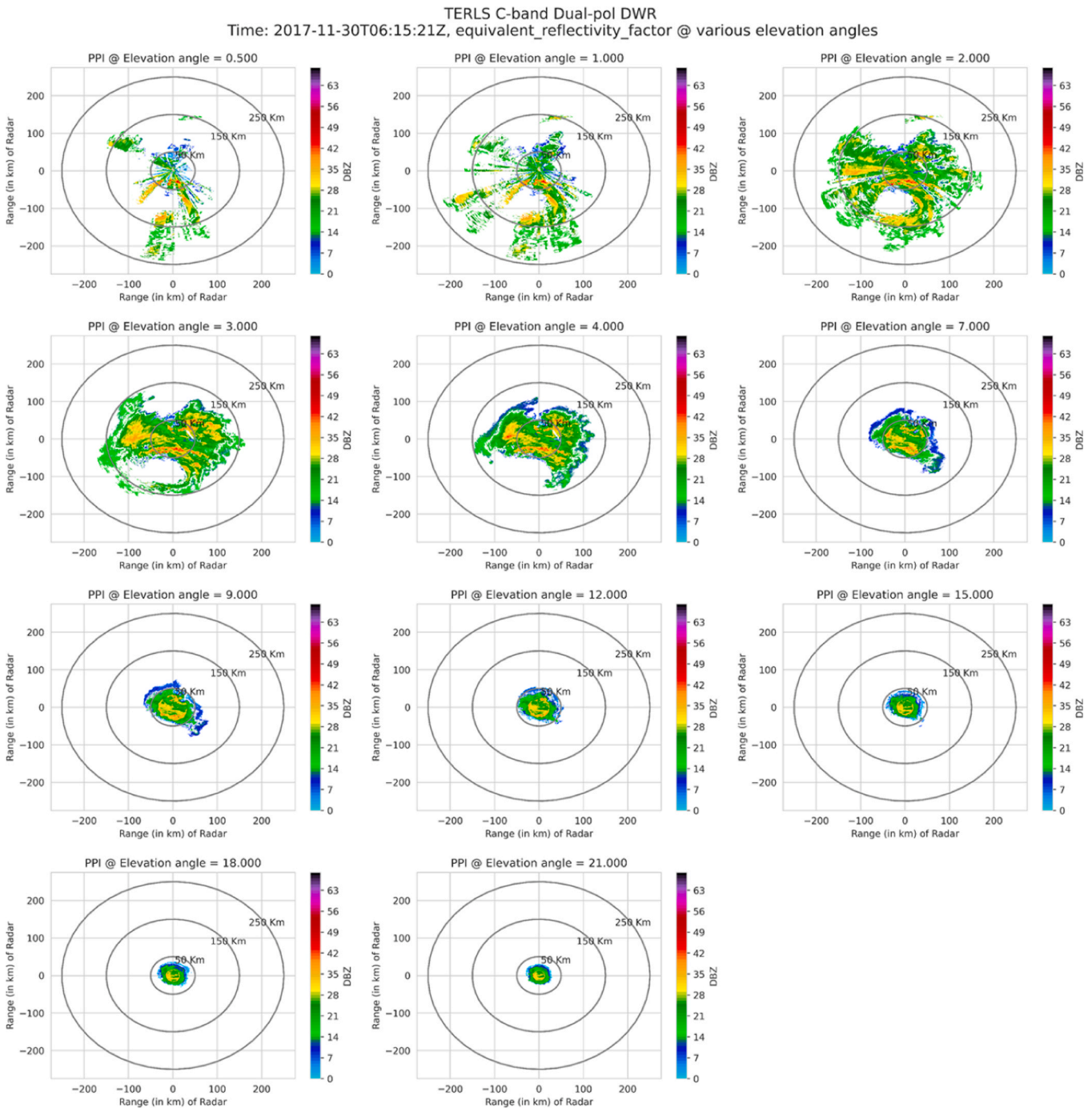

Fig. 6. PPI plot of reflectivity at various elevations for November 30, 2017 06:15:21 UTC.

where r represents the distance between radar to gate center, θ_a denotes the azimuth angle, θ_e stands for the elevation angle, s indicates the arc length, and R represents the effective radius of the Earth, which is considered to be 4/3 of the mean radius of the Earth (6371 km). The transformation of cartesian coordinates (x, y) to geographic coordinate system (lat, lon) is performed using Eqs. 4, 5 and 6 described below [35]:

$$lat = \arcsin(\cos(c)\sin(lat_o) + \frac{ysin(c)\cos(lat_o)}{\rho}) \tag{4}$$

$$lon = lon_o + \arctan2(\mathrm{xsin}(c),\ \rho\cos(lat_o)\cos(c) - ysin(lat_o)\sin(c)) \tag{5}$$

$$\rho = \sqrt{x^2 + y^2}\ ;\quad \mathrm{c} = \rho \Big/ R \tag{6}$$

where x, y are the cartesian position, lat, lon are corresponding latitude and longitude and lat_o, lon_o are the latitude and longitude of the center of the projection; R is the radius of the earth. "*sweeps2mergednc*" function helps in merging cfradial (polar) data from IMD radars that contain all 10/11 sweeps from single scans. This function is capable of storing single or dual polarization radar files in both polar as well as gridded radar data.

The function takes in parameters like filename, grid shape (altitude levels, x-axis grids, and y-axis grids), and height in km to be considered for making grid levels for altitude and length of radar range. The simplicity of this function is that it can directly make grid objects for the user from any NetCDF file, either file which is the product of "*raw2nc*",

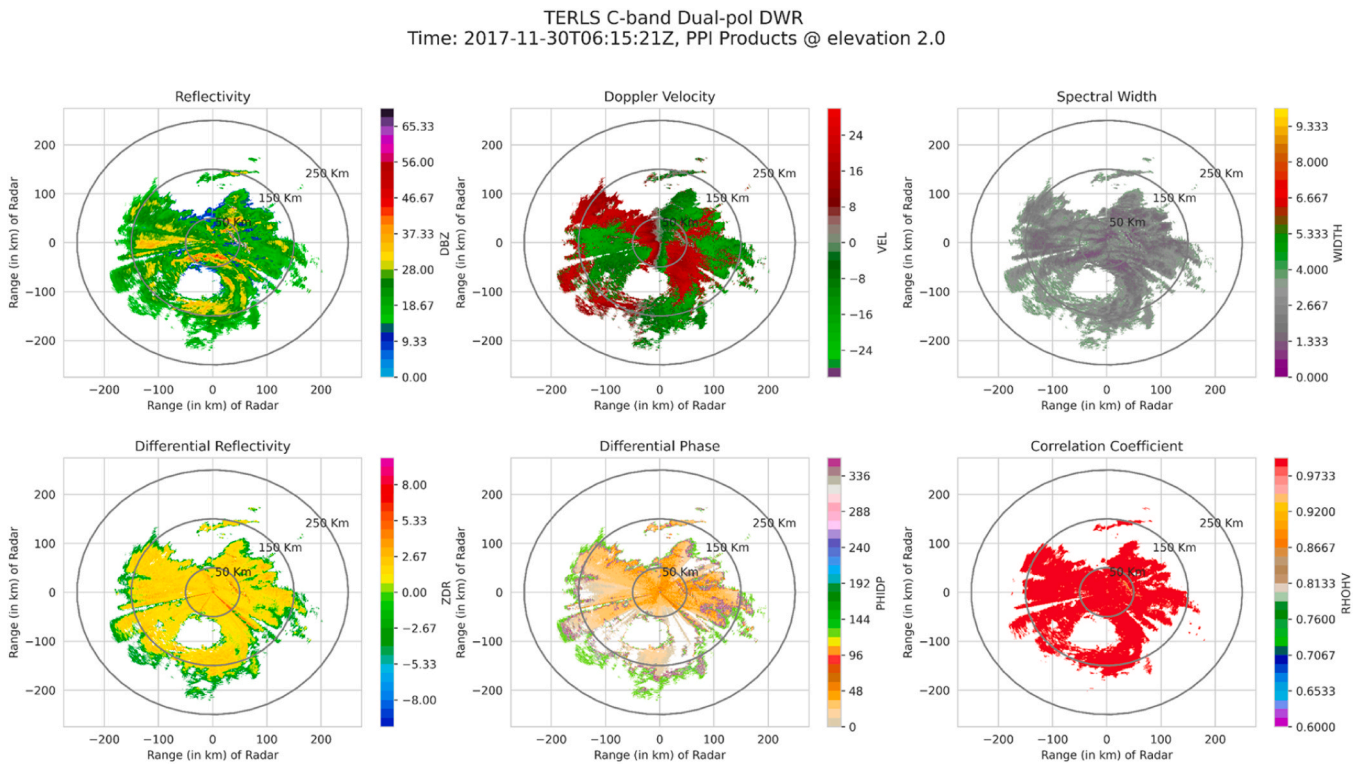

Fig. 7. Plot of various fields including reflectivity, radial velocity, spectral width, differential reflectivity, differential phase, correlation coefficient showing spatial of "Ockhi" November 30, 2017 at 06:15:21 UTC at 2° elevation.

"*format_correctednc*" or DWR NetCDF file from any source which might have missing parameters and attributes in it. All three functions have an added advantage that they are specifically designed in such a way that it can be used for input-output reading and writing DWR data offering flexibility and convenience in data storage and sharing. Listing 1 demonstrates the usage of key functions for format transformation and data reading.

4.2. *analysis.py*

This module offers various functions like "*cappi*", "*cappi_max*", "*marginal_max*", and "*marginal_max_map*", written minding the intended visualization for gridded data in cartesian. A CAPPI (Constant Altitude PPI) is a flat, two-dimensional depiction of radar data at a specific altitude level. It is derived by interpolating data from multiple PPI (Plan Position Indicator) scans taken at various elevation angles. To generate the reflectivity recorded on a plane at a constant height, those fragments of radar field information of the different elevations that are closer to the height for which the CAPPI is to be generated are used. Listing 2 demonstrates the usage of visualization functions for Constant Altitude Plan Position Indicator (CAPPI) plots. The functions like "cappi" and "cappi_max", enabling users to plot CAPPI (for any altitude level) and MAX CAPPI for any specific radar field (DBZ/VEL/WIDTH /ZDR/PHIDP/RHOHV) for any radar location (SHAR/TERLS/CHERRAPUNJI). The "marginal_max" function adds top and right marginal cross-sections to the MAX CAPPI plot for vertical insights in cartesian plane while "marginal_max_map" allows adding different base map styles with an option to choose the background in geo-referenced mapped plane. Users can customize plots with optional parameters like grid, rings, ticks in km and option for saving the image as defined in the function definition. Moreover, the "*analysis.py*" module offers radar reflectivity correction function named "*corr_ref*" which utilizes the correction algorithms using polarimetric variable [32]. A combination of the Gabella filter and a fuzzy-logic based radar echo classification has been used for clutter identification and to identify non-meteorological echoes, this identification is based on the fact that the non-meteorological echoes decorrelate rapidly in space and are spatially heterogeneous [7,32]. The filtering algorithm, in itself, is divided into two parts: the first part is a "spatial proximity" filter and the second is a test of compactness. The "*cfad*" function (Contour Frequency Altitude Diagram) display the contour for the frequency distribution with respect to the altitude. The "*timeseries_spatial*" function gives the time series plots for storm tracking, flood analysis and continuos monitoring of heavy rain events by providing range of radar files for which the time series is visualized. The "*timeseries_location*" gives the rain intensity time series for a location, for the range of radar files at a specific altitude. This module also offers various algorithms for estimating rain rates by using "*qpe_cappi*" function to visualize precipitation data in CAPPI. It allows for the utilization of different Z-R relationships tailored for distinct rain types, such as stratiform and convective rain, based on conditional reflectivity values [5,27]. Additionally, the module encompasses different rain rate estimation techniques, including those that integrate polarimetric measurements, as depicted in Eqs. 7, 8, 9, and 10 and 11.

$$R(Z_h) = (Z_h/a)^{1/b} \tag{7}$$

$$R(Z_h, Z_{dr}) = aZ_h^b Z_{dr}^c \tag{8}$$

$$R(K_{DP}) = aK_{DP}^b \tag{9}$$

$$R(K_{DP}, Z_{dr}) = aK_{DP}^b Z_{dr}^c \tag{10}$$

$$R(K_{DP}, Z_{dr}, Z) = aK_{DP}^b Z_{dr}^c Z^d \tag{11}$$

where a, b,c and d are the coefficients, R represents rain rate in mm h^{-1}, Z_h and Z_{DR} are reflectivity and differential reflectivity value, respectively in dBZ and dB. Z_h is in mm^6 m^{-3}, K_{DP} is in deg km^{-1} [3–5,12,18]. Furthermore, this module supports the utilization of a dual Z-R relationship based on the conditional range of radar reflectivity for stratiform and convective rainfall.

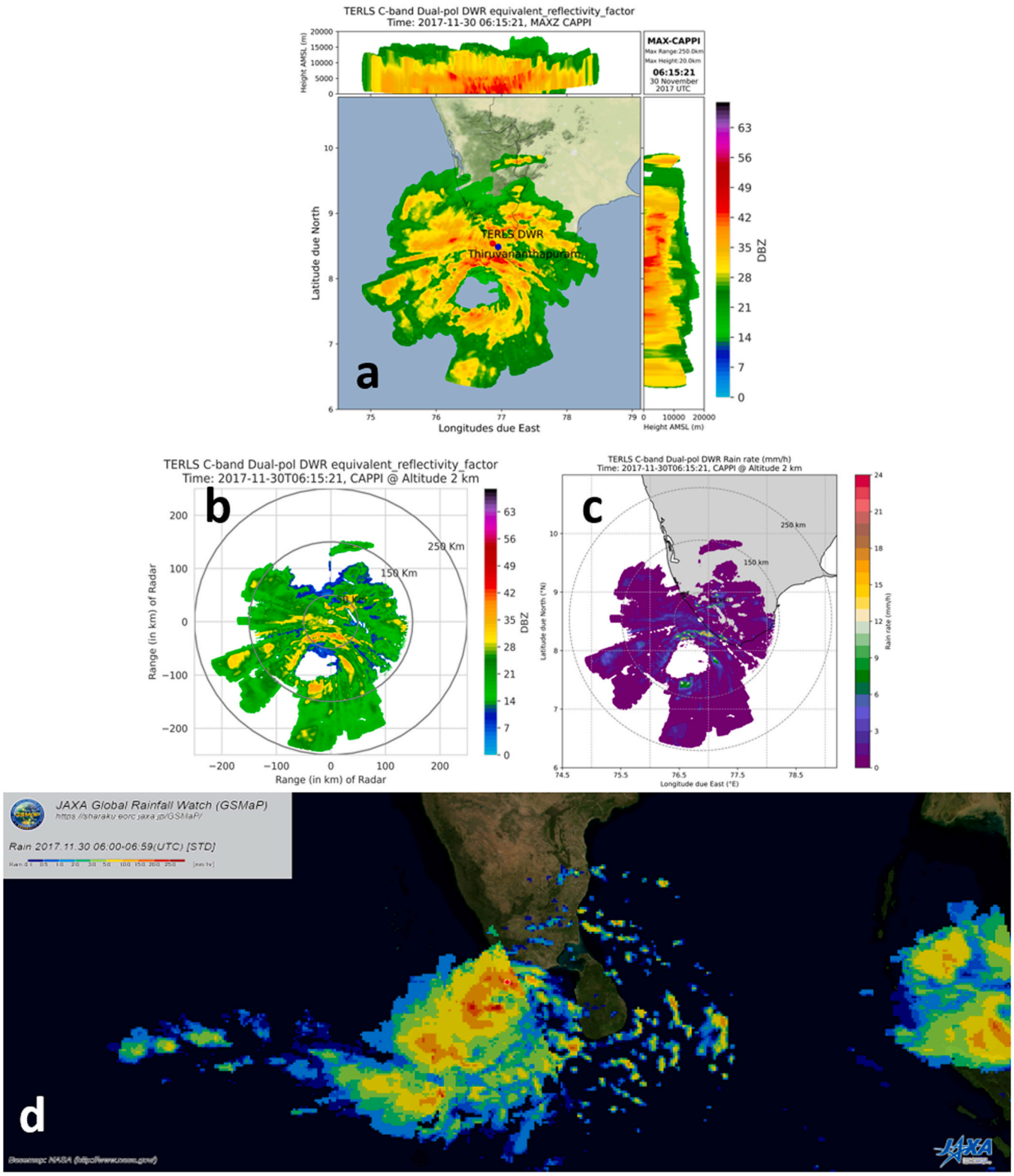

Fig. 8. a.) MAX-Z CAPPI for November 30, 2017 at 06:15:21 UTC, b.) CAPPI at 2 Km altitude c.) corresponding radar QPE and d.) rain rate from 06:00 – 06:59 (UTC) on November 30, 2017 from JAXA GSMaP.

4.3. utilities.py

The "*utilities.py*" module serves as a complement to the visualize, analyze, and transform modules, enhancing their functionality with additional features. It includes functions such as "*fread*", which reads raw radar data, "*raw_product_list*" for writing raw data into a list structure, "*raw_reshape_stack*" for stacking radar data based on various fields and variables, and "*raw2object*" for converting stacked raw data into radar objects. The "*update_xarray_dataset*" function is used to update xarray grids by addressing missing attributes and other data issues. Additionally, the "*extract_start_time*" function aids in correcting time discrepancies in radar data. Other functions like "*db2si*" are responsible for converting

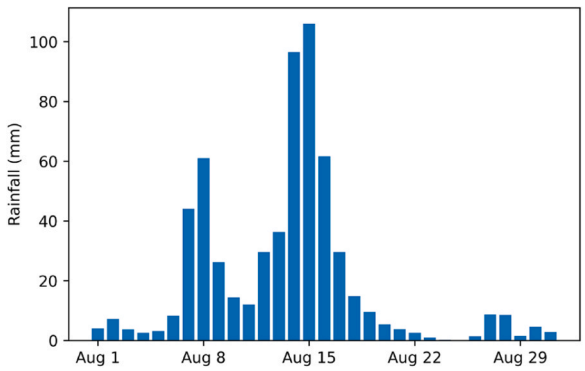

Fig. 9. Accumulated precipitation averaged over Keral region during August 2018 using IMD gridded rainfall data [39].

decibel values into SI units, while "*qpe_estimators*" facilitate the conversion of radar measurements into rainfall estimates using reflectivity, differential reflectivity, and specific differential phase. The "*extract_values_location*" is used to extract rain intensity values for a specific latitude-longitude from multiple radar files creating a rain intensity time series. The "*make_grid*" and "*sorting_key*" are used for making grid from cfradial and sorting multiple sweep files in an order, respectively.

4.4. *visualize.py*

"pyiwr" also provide a variety of basic visualization functions of the radar base and derived products in PPI form using visualize.py module. It offers various functions like "*elevation*", "*all_elevation*" and "*fields_elevation*". The functions "*elevation*" and "*all_elevation*" visualize radar data for any field at single and multiple elevation angles, respectively. The "*fields_elevation*" function plots multiple radar field (DBZ/VEL/WIDTH/ZDR/PHIDP/RHOHV) at a chosen elevation angle. Listing 3 demonstrates the usage of visualization functions for Plan Position Indicator (PPI) plots.

5. Potential uses of pyiwr

Pyiwr, is designed as a specialized tool for processing Indian weather radar data, serving as a pivotal resource for meteorological analysis. Its robust capabilities have been extensively validated during developmental and testing phases. Apart from reading and visualization of the base DWR products, the pyiwr offers various functionalities like gridding of PPI data to CAPPI, processing raw reflectivity data (clutter correction), QPE using Z-R relationships (both single and for different rain types) and other dual polarization products relationships, time-series analysis that can be used in analyzing various meteorological events. This section presents a few case studies highlighting the practical usage of pyiwr.

5.1. *Radar reflectivity correction procedure: A Case for Cherrapunji Radar*

In radar meteorology the primary interest is the radar echoes from the hydrometeors like raindrops, ice crystals, cloud droplets etc. However, all the echoes do not originate from these hydrometeors and are termed as clear air echo. These can be originated from fixed targets like trees, buildings other objects etc, and are termed as ground clutter. As the name suggests these are unwanted signals, contaminate the weather signals and need to be filtered out from the received signal to accurately derive the radar variables. In literature, various methods have been developed that identify clutter based on their properties like narrow spectrum width, close to zero radial velocity and stationarity of the echoes in time. A number of techniques have been developed using these properties of logic-based decision like fuzzy logic [7,32]. The radar data quality control is extensively studied by researcher across the globe. The beam blockage fraction studied for Cherrapunji radar [17] clearly shows in Fig. 3, that to the north-northwest of radar there was significant beam blockage. The topographically complex location of the radar makes it more susceptible to clutter and requires appropriate quality control before using these observations for further processing.

The significant event of Cyclonic Storm that occurred on March 25–26, 2022, is used for present case study. During this period, Cherrapunji experienced a very high rainfall of about 438 mm. The intense rainfall occurred from 09:29 to 13:39 IST on March 26 and continued as a continuous rain shower. This event stood out due to its extraordinary rainfall amount and duration. A MAX-CAPPI plot of reflectivity is shown in Fig. 4a.Various small and isolated areas of the reflectivity can be clearly seen in the Meghalaya region, corresponding to the echoes from the clutter, and it is difficult to distinguish the low intensity rain from such clutter [37]. Thus, to mitigate these effects of echoes from clutter, we have employed a fuzzy-logic filter using radar polarimetric variables [32] using the "*corredted_dbz*" function in *analysis* module of pyiwr. The corrected reflectivity as shown in Fig. 4b significantly removed the non-meteorological echoes. This makes the radar data more reliable for further processing while performingthe meteorological analysis.

5.2. *Radar QPE analysis: A case study of Ockhi cyclone*

In late 2017, the Indian subcontinent faced one of its most devastating storms of the year, the Very Severe Cyclonic Storm (VSCS) Ockhi. Ranking as the third and most potent cyclonic storm in the North Indian Ocean cyclone season of 2017, Ockhi wreaked havoc and resulted in significant damage and loss of life. Originating from a low-pressure area in the southwest region of the Bay of Bengal on November 28, 2017, Ockhi traversed approximately 2500 km, crossing over the Indian Ocean and Arabian Sea before making landfall on the south coast of Gujarat on December 6, 2017 [15].

Polarimetric radars have emerged as pivotal tools for understanding the characteristics of tropical cyclones, as evidenced by numerous studies [33]. This case study of Ockhi cyclone is chosen to highlight the kind of analysis performed with the help of pyiwr toolkit. The data from TERLS, C-band radar stationed at Thumba Equatorial Rocket Launching Station (TERLS) in Thiruvananthapuram is used in the case study for November 30, 2017.

Firstly, in order to understand the development, progress and spatial extent of the cyclone the time series of radar reflectivity was generated using the "*timeseries_spatial*" function available in "*analysis*" module. The time series of DWR reflectivity images (after every ~6 hour) as shown in Fig. 5 provides insight into the progression of the 'Ockhi' cyclone as it traversed through various stages from November 30, 2017, 00:15 UTC to 23:15 UTC. Progressing in a westward direction, the cyclone underwent intensification, evolving into a mature cyclonic structure. By 06:15 UTC, the DWR successfully discerned the cyclone's well-defined eye as shown in Fig. 5. During this interval, the cyclone's center was positioned approximately 20 km off the Thumba coastline [26]. Fig. 6 shows the spatial variation of the cyclone's reflectivity at different elevations, providing valuable insights into the cyclone's structure at different elevation, notably, at a 2-degree elevation, the DWR clearly captures both the cyclone's overall structure and the presence of its central eye.

Apart from analysis of the spatial extent and progression of the cyclone using radar reflectivity, the radar dual products also provide valuable information of the size and orientation of the hydrometeors. Fig. 7 illustrates the spatial distribution of various base fields and dual-pol fields, providing insights into the key features of the cyclone. Positive values of Zdr indicate horizontally oriented hydrometeors, while negative values denote vertically oriented ones. Near-zero values of Zdr typically indicate spherical hydrometeors.

The MAX-Z CAPPI plot is shown in Fig. 8a derived from the gridded data sets generated using "*sweep2gridnc*" function. It distinctly reveals the cyclone's eye, inner, and outer rainbands with highest observed

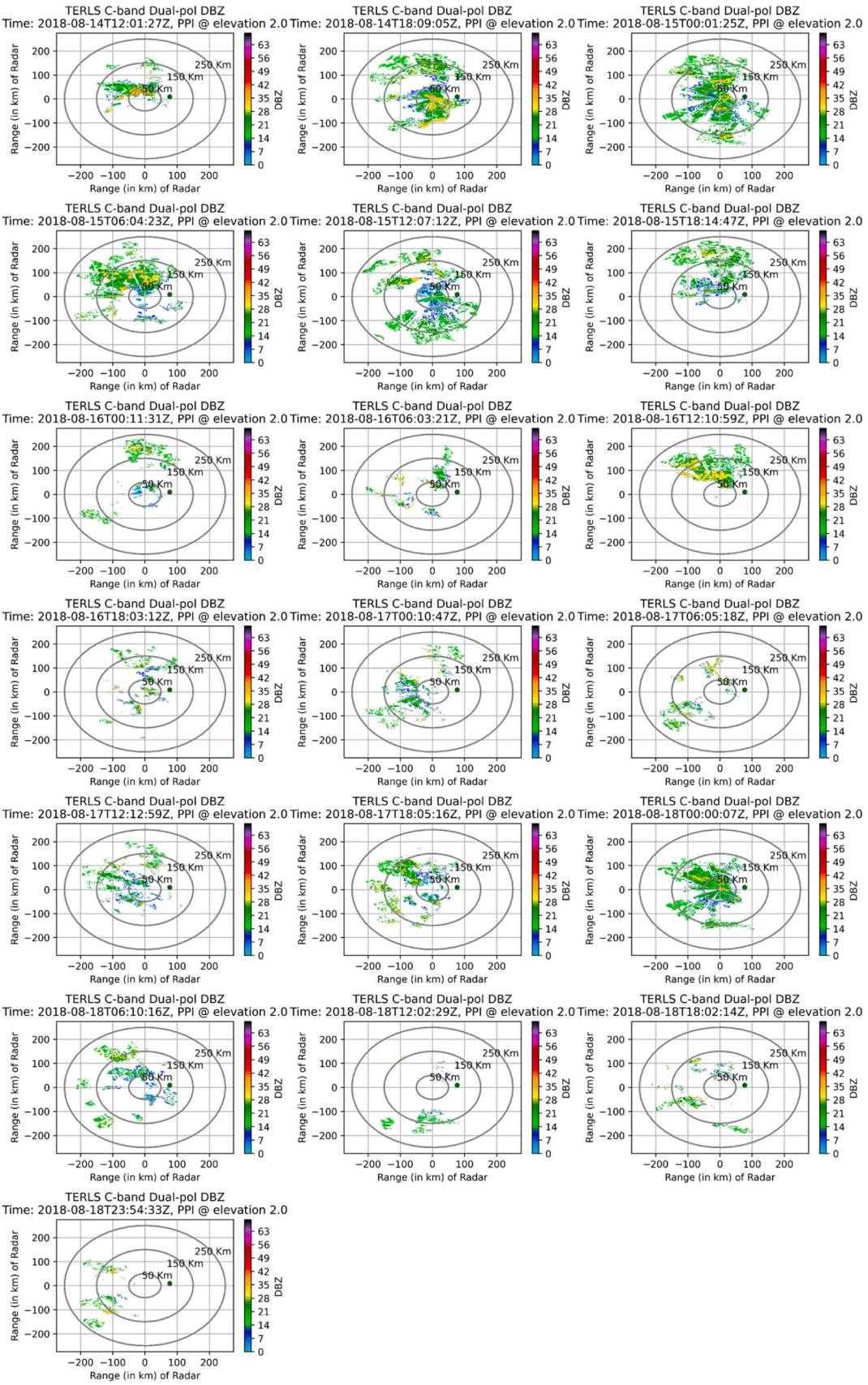

Fig. 10. 6 hourly time series plot of DWR reflectivity during kerala floods from August 14, 2018 12:01:27 UTC to August 18, 2018 23:54:33 UTC at 2° elevation.

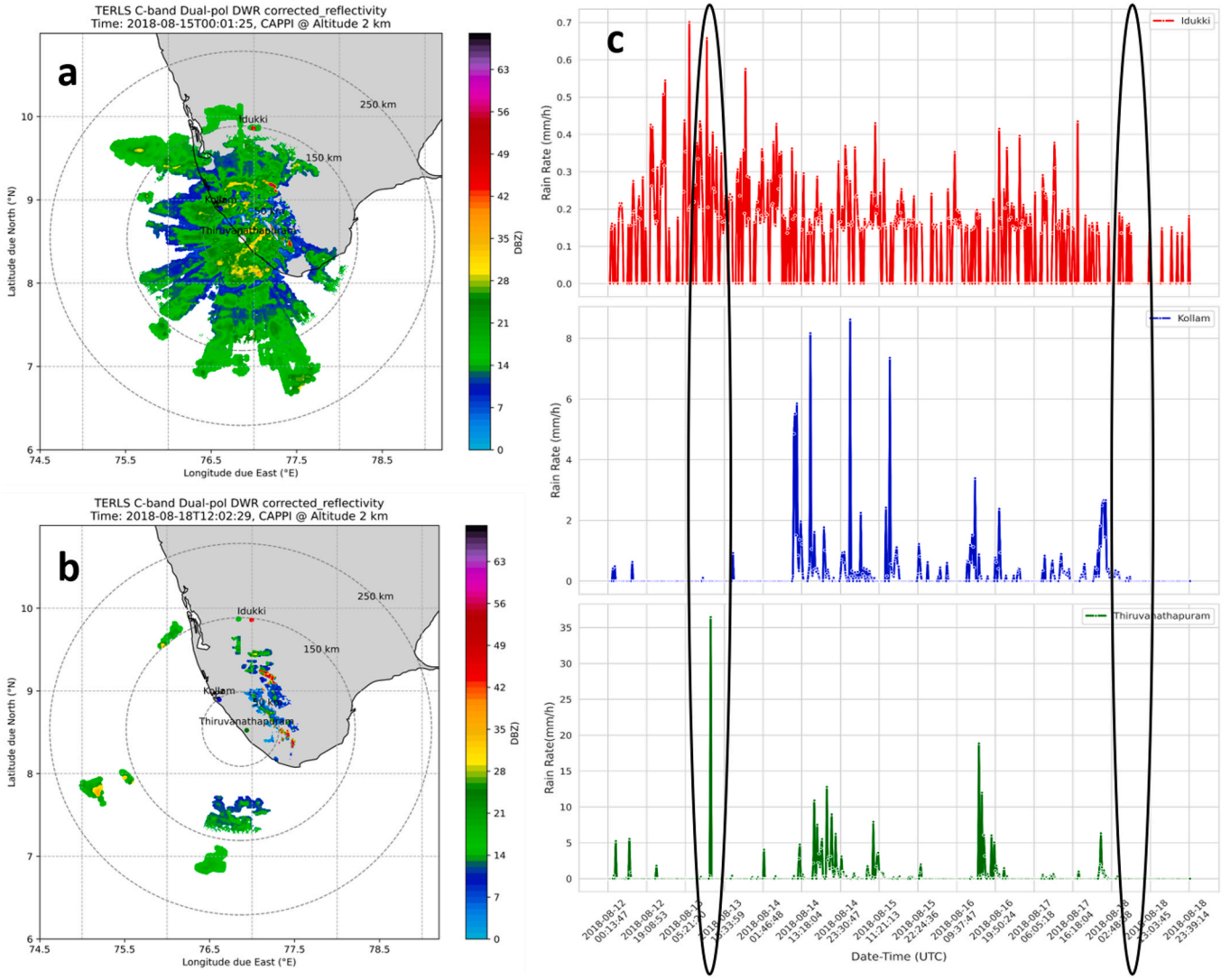

Fig. 11. a.) CAPPI at 2 Km for August 15, 2018 00:01:25 UTC, b.) CAPPI at 2 Km for August 18, 2018 12:02:29 UTC showing red dot for Idukki, blue dot Kollam and green dot for Thiruvananthapuram, c.) Estimated rain intensity time series plot at color marked locations labeled in the CAPPI (a and b plot), black markings shows the rain intensity corrosponding to each CAPPI respectively.

Listing 1: Format transform and reading data as radar object / Xarray grid

```
#For dual -pol raw dwr files
radar = pyiwr.transform.raw2nc (filename, save_file =True)
#For mosdac nc files
radar = pyiwr.transform.format_correctednc (filename, save_file =True)
#For xarray grids for NetCDF files
xarray_grids = pyiwr.transform.sweeps2gridnc (filename, grid_shape = (81, 501, 501), height =20, length =250, save_file =True)
#For merging multiple sweep files into a single file for making volumetric or gridded NetCDF file
Radar_grids = pyiwr.transform.sweeps2mergednc (path_string, start_index = 0, end_index=None, scan_type="B", dualpol=False, gridded=False, save_file=False)
```

reflectivity intensity, reaching around 45–50 dBZ. Overshooting clouds characterized by reflectivity values Z ≥ 20 dBZ was observed. Region of high reflectivity values along with the corresponding high values of precipitation with rain rate 15–20 mm/hr can be seen in Fig. 8d. In order to estimate the rain rate from the radar reflectivity the "*qpe_cappi*" function of "*analysis*" module is used. The user can define the customize a and b constant for the Z-R relation based QPE and the desired altitude as well. By default, the Z-R relationship used is "a = 267" and "b =1.3" at 2 km altitude as operationally provided as Surface Rainfall Intensity (SRI) by the Meteorological & Oceanographic Satellite Data Archival Centre (MOSDAC), Space Application Centre – Indian Space Research Organization (SAC-ISRO). Fig. 8b shows the reflectivity CAPPI value at 2 km altitude and the corresponding rain rate is shown in Fig. 8c. The analysis described above highlighted the practical usage and

Listing 2: Analysis and Visualization for Reflectivity Correction, CFAD, Time-series and Radar QPE

```
#For CAPPI in cartesian plane.
pyiwr.analisys.cappi(xarray_grids, altitude_level =3, field_name ='DBZ', radar_location ='CHERRAPUNJI', grid=False, rings =False, colorbar_range=[0,70], save_image =True, img_name ='img.png')
#For MAX CAPPI in cartesian plane
pyiwr.analisys.cappi_max(xarray_grids, field_name ='DBZ', radar_location ='CHERRAPUNJI', grid=False, rings =False, colorbar_range=[0,70], save_image =True, img_name ='img.png')
#For MAX CAPPI (with marginal cross-sections) in cartesian plane
pyiwr.analisys.marginal_max(xarray_grids, radar_location ='SHAR', field_name ='DBZ', show_rings =True, show_grid =True, show_cross_sections =True, save_image =True, img_name ='img.png')
#For MAX CAPPI (with marginal cross-sections) with map
pyiwr.analisys.marginal_max_map(xarray_grids, radar_location ='TERLS', field_name ='DBZ', background ='terrain-background', cross_sections =True, save_image =True, img_name ='img.png')
#For radar QPE CAPPI.
pyiwr.analisys.qpe_cappi(xarray_grids, altitude_level =2, field_name ='DBZ', radar_location ='CHERRAPUNJI', grid=False, rings =False, cartesian=False, colorbar_range=[0,70], type = 'ZR', a = 276, b = 1.3, c = None, d = None, a_c= None, b_c= None, a_s= None, b_s= None, save_image =True, img_name ='img.png')
#For spatial PPI time-series.
pyiwr.analisys.timeseries_spatial(folder_path, start_index=0, end_index=None, field_name='DBZ', elevation_index=0, rings=True, grid=True, colorbar_range=None, save_image=False, img_name=None)
#For location-based time-series for QPE.
pyiwr.analisys.timeseries_location(folder_path, start_index=0, end_index=None, field_name='DBZ', altitude_level=2, type = 'ZR', a = 276, b = 1.3, c = None, d = None, a_c= None, b_c= None, a_s= None, b_s= None, lat=8.5241, lon=76.9366, place_name='Thiruvanathapuram', save_image=False, img_name=None)
#For corrected reflectivity.
pyiwr.analisys.corr_ref filename, save_file =True)
#For CFAD plot
pyiwr.analisys.cfad(xarray_grids, field_name='DBZ', save_image =True, img_name ='img.png')
```

Listing 3: Visualization of Basic Radar fields

```
#For PPI at specific elevation
pyiwr.visualize.elevation(radar, field_name='DBZ', elevation_index=0, rings=True, grid=True, colorbar_range=[0,70], save_image=True, img_name='img.png')
#For PPI of any field at all elevation
pyiwr.visualize.all_elevation(radar, field_name='DBZ', rings=True, grid=True, colorbar_range=[0,70], save_image= True, img_name='img.png')
#For PPI of all fields
pyiwr.visualize.fields_elevation(radar, elevation_index=0, colorbar_range=[0,70], rings=True, grid=True, save_image= True, img_name='img.png')
```

applicability of the pyiwr in cyclone studies.

5.3. Time series analysis: A case of 2018 Kerala floods

Flooding stands out as a major natural peril, with the potential to cause significant loss of life and severe socio-economic repercussions [11]. In recent years, the incidence of flooding has surged, attributed to global population expansion and the effects of climate change [36]. While in-situ rain gauges offer precise surface precipitation data, they often fall short in capturing spatial and temporal variability. Weather radar, on the other hand, furnishes continuous data spanning vast areas, even in regions lacking dense gauge point observation networks, thus providing a valuable resource to mitigate this limitation.

Radar's capacity to capture the spatial distribution of precipitation has positioned it as a pivotal tool in meteorological research [6]. A case study of devastating floods that struck the state of Kerala, India, in August 2018 is presented, leveraging C-band polarimetric DWR data at Thumba (8.5°N, 77°E), India. As per the analysis of IMD daily grided rainfall, Fig. 9 shows the peak rainfall occurs around August 15. Studies suggested that, heavy rainfall occurred in two spells from August 7–10 and 14–18, with the highest recorded rainfall occurring on August 9, 14 and 15 [28,31]. Fig. 10 shows the 6 hourly time series of the radar reflectivity PPI at 2-degree elevation, showing the evolution and movement of the convective system. It clearly highlights that around August 15, 2018 00:01:25 UTC, various parts of the states experienced continuous rainfall as seen in Fig. 11a. This event persisted till August 18. 2018 as seen in Fig. 11b where the precipitation was limited to certain places in Kerala only. Fig. 11c shows the time series of the whole event duration at the three locations namely Idukki, Kollam and Thiruvananthapuram from August 12-18, 2018.

6. Discussion and outlook

The presented research introduces the Python Indian Weather Radar Toolkit (pyiwr), an open-source Python module specifically designed to address the challenges associated with Indian Weather Radar data analysis. The main problem solved by pyiwr is the redefining of raw format of Indian DWR files into standard radar object NetCDF files and restructuring DWR radar NetCDF files with date-time correction issue, missing sweep ray index and metadata correction, enabling standardized data representation and compatibility with various popular Python libraries like Py-ART and wradlib. Moreover, pyiwr is fully compatible with the ISRO raw DWR file formats, MOSDAC open DWR data for TERLS, SHAR, and CHERRAPUNJI radars, IMD radars and Indian

Listing 4: Installation instructions

```
# Create a new conda env named 'pyiwr'
conda create -n pyiwr python=3.9/3.10/3.11 jupyter git -c conda-forge
# Activate the 'pyiwr' env
conda activate pyiwr
# Install 'pyiwr' package from GitHub using pip
pip install git+https://github.com/nitigsingh/pyiwr.git
```

Institute of Tropical Meteorology (IITM) radar systems, allowing users to effortlessly work with data from any radar systems in India. Another major contribution for pyiwr, is the merging of multiple elevation files user gets in the IMD radar systems, into single volumetric file containing multiple elevation for a single radar scan. This compatibility broadens the scope of radar data analysis and enhances the toolkit's versatility for various research applications. The module also offers a wide range of analyzation and visualization functions, namely radar QPE, timeseries analysis and radar data quality control, catering to various polarimetric products to gain meaningful insights from the radar data, facilitating a deeper understanding of the analyzed weather patterns and other systems.

Potential and Further Development: The pyiwr module is designed to provide users with a user-friendly experience, complemented by detailed and comprehensive documentation. This allows users to efficiently implement various algorithms, saving valuable time and effort. Version v1.0.0 of the pyiwr, an open-source distribution and processing program has been accessible since July, 2023, the developers are actively working to ensure the availability of the package through the pip package manager and the Anaconda distribution as well. As an active project, pyiwr intends to increase its capabilities by including other data sources and formats, as well as continuously improving flexibility in the visualization tools and seeks contributions from the research community, making it a dynamic and collaborative toolkit for Indian weather radar research and applications. Future work includes, incorporating some advanced data processing algorithms like radar signal attenuation correction, precipitation type classifications, melting layer identification and radar calibration etc. Inclusion of these function in pyiwr, would be beneficial in order to create the most comprehensive and powerful openly available set of tools for the Indian weather radar community working in the applications of radar meteorology in the atmospheric, climate, hydrology, ocean, and earth sciences.

CRediT authorship contribution statement

Nitig Singh: Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Vaibhav Tyagi:** Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Saurabh Das:** Conceptualization, Resources, Supervision, Writing – review & editing. **Udaya Kumar Sahoo:** Conceptualization, Data curation, Resources, Writing – review & editing. **Shyam Sundar Kundu:** Conceptualization, Resources, Supervision, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The author has shared the link to code repositories in the manuscript.

Acknowledgments

We sincerely acknowledge North Eastern Space Applications Centre (NESAC, https://nesac.gov.in) for providing us with CHERRAPUNJI raw DWR data and to Meteorological & Oceanographic Satellite Data Archival Centre (MOSDAC/SAC/ISRO, https://mosdac.gov.in) for providing open DWR data as NetCDF. One of the authors (SD) also thankfully acknowledges the financial support received under SERB (CRG/2022/006986) and MoES NARM(Moes/16/04/2021-RDESS/NARM-4) program and authors (SD, NS and VT) also thankfully acknowledges the financial support received under OceanSat-03 utilization program.

Code Availability Section

Pyiwr's development and project hosting is done on GitHub, utilizing its version control system for source code changes, bug tracking, feature requests, and documentation hosting for the project. The library is in its heavy developmental phase, so contributions are welcome. The pyiwr team will provide support on the integration or porting of the code, if necessary. Contributions are made through GitHub pull requests. Pyiwr is also available on Zenodo, a research data repository with a permanent digital identifier for shared research outputs.

Operating system: Linux and Windows
Programming language: Python 3. (3.9, 3.10 and 3.11 tested)
Additional system requirements: None
Dependencies: NumPy, matplotlib, netcdf4, cartopy xarray and pyart
List of developers: Nitig Singh and Vaibhav Tyagi, SIGMA Research Laboratory, IIT Indore, India
Software Location:
Archive
Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.8192061
License: MIT
Publisher: Singh, Nitig
Version published: v1.0.0
Rolling out version = v2.0.0
Date published: 28 July 2023
Code repository
Name: GitHub
Persistent identifier: https://github.com/nitigsingh/pyiwr.git
License: MIT
Date published: 28 July 2023
Language: English

Appendix

Listing 4 demonstrate instructions how to set up the required environment and install the pyiwr package from the GitHub repository. For more comprehensive instructions on effectively utilizing pyiwr including detailed breakdown of the library's functionalities, usage example notebooks, and practical guidelines, please access the documentation provided at the following web address: https://nitigsingh.github.io/pyiwr.

References

[1] D. Atlas, Radar calibration: some simple approaches, Bull. Am. Meteorol. Soc. 83 (9) (2002) 1313–1316, https://doi.org/10.1175/1520-0477-83.9.1313.

[2] S.K.R. Bhowmik, S.S. Roy, K. Srivastava, et al., Processing of Indian doppler weather radar data for mesoscale applications, Meteorol. Atmos. Phys. 111 (2011) 133–147, https://doi.org/10.1007/s00703-010-0120-x.

[3] V.N. Bringi, V. Chandrasekar, Polarimetric Doppler Weather Radar: Principles and Applications, Cambridge University Press, 2001, https://doi.org/10.1017/CBO9780511541094.

[4] V.N. Bringi, T.D. Keenan, V. Chandrasekar, Correcting C-band radar reflectivity and differential reflectivity data for rain attenuation: A self-consistent method with constraints, IEEE Trans. Geosci. Remote Sens. 39 (9) (2001) 1906–1915, https://doi.org/10.1109/36.951081.

[5] V.N. Bringi, M.A. Rico-Ramirez, M. Thurai, Rainfall estimation with an operational polarimetric C-band radar in the United Kingdom: comparison with a gauge network and error analysis, J. Hydrometeorol. 12 (5) (2011) 935–954, https://doi.org/10.1175/JHM-D-10-05013.1.

[6] R.J. Doviak, D.S. Zrnic, Doppler radar and weather observations. Dover Books on Engineering Series, Dover Publications, 2006. ⟨https://books.google.ch/books?id=ispLkPX9n2UC⟩. ISBN 9780486450605. URL.

[7] M. Gabella, J. Joss, G. Perona, G. Galli, Accuracy of rainfall estimates by two radars in the same Alpine environment using gage adjustment, J. Geophys. Res. 106 (D6) (2001) 5139–5150, https://doi.org/10.1029/2000JD900487.

[8] J.J. Helmus, S.M. Collis, The python arm radar toolkit (py-art), a library for working with weather radar data in the python programming language, J. Open Res. Softw. 4 (2016), https://doi.org/10.5334/jors.119.

[9] M. Heistermann, S. Jacobi, T. Pfaff, Technical note: an open source library for processing weather radar data (wradlib), Hydrol. Earth Syst. Sci. 17 (2013) 863–871, https://doi.org/10.5194/hess-17-863-2013.

[10] J.D. Hunter, Matplotlib: A 2D graphics environment, Comput. Sci. Eng. 9 (03) (2007) 90–95.

[11] S.N. Jonkman, J.K. Vrijling, Loss of life due to floods, J. Flood Risk Manag. 1 (1) (2008) 43–56.

[12] J.S. Marshall, W.M.K. Palmer, The distribution of raindrops with size, J. Meteor. 5 (4) (1948) 165–166, https://doi.org/10.1175/1520-0469(1948)005<0165:TDORWS>2.0.CO;2.

[13] Met Office, 2010 - 2015. Cartopy: a cartographic python library with a Matplotlib interface. ⟨https://scitools.org.uk/cartopy⟩.

[14] K.J. Millman, M. Aivazis, Python for scientists and engineers (URL), Comput. Sci. Eng. 13 (2) (2011) 9–12, https://doi.org/10.1109/MCSE.2011.36, ⟨https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.36⟩ (URL).

[15] A. Nair, S.S. Das, A. Thomas, C. Sarangi, V.P. Kanawade, Role of Cyclone "Ockhi" in the re-distribution of aerosols and its impact on the precipitation over the Arabian Sea, Atmos. Res. 235 (2020) 104797.

[16] Numpydoc, 2015. Numpy's sphinx extensions. ⟨https://github.com/numpy/numpydoc⟩.

[17] Sambit Panda, 2019, Quality control and Post-processing of ISRO Doppler Weather Radars. 10.13140/RG.2.2.14909.49127.

[18] A.V. Ryzhkov, D.S. Zrnic, Radar Polarimetry for Weather Observations. Springer Atmospheric Sciences, Springer International Publishing, 2019, https://doi.org/10.1007/978-3-030-05093-1.

[19] A.V. Ryzhkov, S.E. Giangrande, V.M. Melnikov, T.J. Schuur, Calibration issues of dual-polarization radar measurements, J. Atmos. Ocean. Technol. 22 (8) (2005) 1138–1155, https://doi.org/10.1175/JTECH1772.1.

[20] A.V. Ryzhkov, T.J. Schuur, D.W. Burgess, P.L. Heinselman, S.E. Giangrande, D. S. Zrnic, The joint polarization experiment: Polarimetric rainfall measurements and hydrometeor classification, Bull. Am. Meteorol. Soc. 86 (6) (2005) 809–824, https://doi.org/10.1175/BAMS-86-6-809.

[21] A.V. Ryzhkov, D. Zrnić, Assessment of rainfall measurement that uses specific differential phase, J. Appl. Meteor. 35 (11) (1996) 2080–2090, https://doi.org/10.1175/1520-0450(1996)035<2080:AORMTU>2.0.CO;2.

[22] D. Sanchez-R, M.A. Rico-Ramirez, Towerpy: an open-source toolbox for processing polarimetric weather radar data, Environ. Model. Softw. 167 (2023) 105746, https://doi.org/10.1016/j.envsoft.2023.105746.

[23] M.F. Sanner, et al., Python: A programming language for software integration and development, J. Mol. Graph. Model. 17 (1) (1999) 57–61.

[25] Stephan Hoyer, Joe Hamman, xarray: Nd labeled arrays and datasets in python, J. Open Res. Softw. 5 (1) (2017).

[26] K.V. Subrahmanyam, S.R. Baby, C-band Doppler weather radar observations during the passage of tropical cyclone 'Ockhi, Nat. Hazards 104 (2020) 2197–2211, https://doi.org/10.1007/s11069-020-04268-2.

[27] N. Singh, V. Tyagi, S. Das, U.K. Sahoo, S.S. Kundu, Investigation of orographic Z-R relationships for three locations in India, 2023 XXXVth Gen. Assem. Sci. Symp. . Int. Union Radio Sci. (URSI GASS), Sapporo, Jpn. (2023) 1–4, https://doi.org/10.23919/URSIGASS57860.2023.10265609.

[28] V. Tyagi, S. Das, S.K. Panda, B.P. Shukla, Investigating the role of horizontal winds in understanding extreme rainfall events: 2018 Kerala floods case study, 2023 IEEE India Geosci. Remote Sens. Symp. (InGARSS), Bangalore, India (2023) 1–4, https://doi.org/10.1109/InGARSS59135.2023.10490372.

[29] Unidata, Network common data form (netcdf). DOI: http://doi.org/10.5065/D6H70CW6.

[30] J.F. i Ventura, M. Lainer, Z. Schauwecker, J. Grazioli, U. Germann, Pyrad: a real-time weather radar data processing framework based on Py-ART, J. Open Res. Softw. 8 (1) (2020) 28, https://doi.org/10.5334/jors.330.

[31] Y. Viswanadhapalli, C.V. Srinivas, G. Basha, H.P. Dasari, S. Langodan, M. Venkat Ratnam, I. Hoteit, A diagnostic study of extreme precipitation over Kerala during August 2018, Atmos. Sci. Lett. 20 (12) (2019) e941.

[32] G. Vulpiani, P. Tabary, J. Parent du Chatelet, F.S. Marzano, Comparison of advanced radar polarimetric techniques for operational attenuation correction at C band, J. Atmos. Ocean. Technol. 25 (2008) 1118–1135, https://doi.org/10.1175/2007JTECHA936.1.

[33] M. Wang, K. Zhao, M. Xue, G. Zhang, S. Liu, L. Wen, G. Chen, Precipitation microphysics characteristics of a Typhoon Matmo (2014) rainband after landfall over eastern China based on polarimetric radar observations, J. Geophys. Res.: Atmos. 121 (20) (2016) 12–415.

[34] Whitaker, Netcdf4 api documentation. ⟨http://unidata.github.io/netcdf4-python/⟩.

[35] D. Zrnić, R. Doviak, G. Zhang, A. Ryzhkov, Bias in differential reflectivity due to cross coupling through the radiation patterns of polarimetric weather radars, J. Atmos. Ocean. Technol. 27 (10) (2010) 1624–1637.

[36] Q. Zhang, X. Gu, V.P. Singh, P. Shi, P. Sun, More frequent flooding? Changes in flood frequency in the Pearl River basin, China, since 1951 and over the past 1000 years, Hydrol. Earth Syst. Sci. 22 (5) (2018) 2637–2653.

[37] N. Singh, S. Das, U.K. Sahoo, S.S. Kundu, S. Chakraborty, On collisional drop breakup in orographic rain, J. Atmos. Res., Volume 300,2024,107232,ISSN 0169-8095, https://doi.org/10.1016/j.atmosres.2024.107232.

[38] H.A. Syed, I. Sayyed, M.C.R. Kalapureddy, K.K. Grandhi, Pyscancf – The open-source Python-based library for IMD Doppler weather radar datasets. Seventh WMO International Workshop on Monsoons (IWM-7). https://doi.org/10.5281/zenodo.5574160.

[39] Pai D.S., Latha Sridhar, Rajeevan M., Sreejith O.P., Satbhai N.S. and Mukhopadhyay B., 2014: Development of a new high spatial resolution (0.25° X 0.25°)Long period (1901-2010) daily gridded rainfall data set over India and its comparison with existing data sets over the region; MAUSAM, 65, 1(January 2014), pp1-18.

Mr. Nitig Singh is an active Project Research Fellow in an ISRO Project at IIT Indore, specializing in Atmospheric & Space Sciences, AI & ML for Climate Informatics, Remote Sensing, Radar Meteorology, Rain Microphysics, Satellite-Based Navigation, and Drone Operations. He holds a Master's degree in Space Science & Engineering from IIT Indore, a Bachelor's degree in Aeronautical Engineering from The Aeronautical Society of India and Diploma in Electrical & Electronics Engineering from Anna University, Chennai. With an interdisciplinary background that includes past roles as a Research Intern at NESAC (ISRO) and a Trainee Engineer at ITR (DRDO), Nitig brings extensive experience in the field of space science and technology. He is a Graduate member of The Aeronautical Society of India and an active member of the IEEE Geoscience and Remote Sensing Society (GRSS). Passionate about exploring innovative solutions in atmospheric and space sciences, Nitig is dedicated to contributing to advancements in these domains.