PyBIRALES: A Radar data processing backend for the real-time detection of space debris

D. Cutajar[1,2], A. Magro[2], J. Borg[2], K. Z. Adami[2,3], G. Bianchi[4], G. Pupillo[4], A. Mattana[4], G. Naldi[4], C. Bortolotti[4], F. Perini[4], L. Lama[4], M. Schiaffino[4], M. Roma[4], A. Maccaferri[4], P. Di Lizia[5], M. Massari[5], M. Losacco[5]
[2] *Institute of Space Sciences and Astronomy (ISSA), University of Malta, Malta*
[3] *Dept. of Physics, University of Oxford, Oxford, OX1 3RH, United Kingdom*
[4] *Istituto Nazionale di Astrofisica - Istituto di Radioastronomia, Via P. Gobetti 101 - 40129 Bologna, Italy*
[5] *Department of Aerospace Science and Technology, Politecnico di Milano, Via G. La Masa 34 - 20156 Milano, Italy*

The growing population of artificial satellites in near-Earth orbit has made the monitoring of orbital debris objects ever more important. Orbital debris objects pose a threat to these satellites as their orbit cannot be changed in order to avoid a collision. In recent years, the European Space Agency (ESA)'s Space Surveillance & Tracking (SST) programme has been assisting national institutions in the upgrading of their space debris detection and monitoring capabilities. One of the latest such systems within this programme is the BIRALES space surveillance system based in Italy. The receiving antenna is a radio telescope that is made up of 32 receivers which are placed on 8 parabolic cylindrical reflectors of the North-South arm of the Istituto Nazionale di Astrofisica (INAF)'s Northern Cross. This work introduces a new software backend which was developed for this novel space debris sensor. The system was designed to be a fast, highly configurable software backend for the radio telescope's acquisition and processing system and whose monitoring and control can be realised by a simple front-end web-based application. The real-time detection of Resident Space Object (RSO) is an important prerequisite for such a system as it gives the operator an immediate feedback loop on any detections whilst keeping the storage requirements at a minimum given that there is no need to save the raw data. The detection of high-velocity objects is achieved by means of a specially developed data processing pipeline that uses the received raw antenna voltages to generate a number of beams, collectively known as a multi-pixel, that cover the Field of View (FoV) of the instrument. The trajectory of the detected objects is determined by considering the illumination sequence within this multi-pixel. The initial results on known objects represent the first steps in extending the growing network of European SST systems.

Keywords: space situational awareness, radar, remote sensing, real-time data processing

1. Introduction

The launch of the Sputnik-1 satellite in the 1957 marked the start of the space age. Since then, thousands of satellites have been put into orbit around Earth. However, only a fraction of these remain operational. This is either due to a planned decommission or a fatal break-up. Break-ups resulting from system malfunction or in-orbit collisions proliferate the number of inactive hardware in space. Further to this, the space environment includes spacecraft used to place the satellites in orbit, such as rocket bodies, exhaust and dust particles and leaked cooling agents (Klinkrad, 2006). Collectively, these objects are referred to as space or orbital debris.

Large catalogued objects are routinely monitored and their orbit is well known. However, only estimates are available for objects that are less than 10 cm in Low Earth Orbit (LEO) and 1 m in Geostationary Earth Orbit (GEO). In LEO, it is estimated that the number of objects between 5 mm and 10 cm is around 600,000 objects (Bonnal & McKnight, 2016). In fact, the orbital debris population is dominated by smaller objects.

[1] Corresponding author.

Objects orbiting the Earth can reach speeds of up to $8\,\mathrm{km\,s^{-1}}$ in LEO. At these speeds, even the smallest of objects can cause extensive damage to an active satellite.

The increasing risk of orbital collisions has led to a heavy investment in new preventive measures that are meant to mitigate the proliferation of space debris, and thereby minimising the collision risks to active satellites. Collision avoidance manoeuvres can be affected when the risk of a collision becomes higher than acceptable thresholds. However, these can only be realised when the orbital state of both cooperative and uncooperative objects is known accurately. Space agencies amalgamate the output of different sensors in order to detect, track and identify both known and unknown orbiting objects in near-Earth orbit. At present, the space environment is characterised through both space-borne (Grassi *et al.*, 2015) and ground-based optical and radar systems. The monitoring of these objects is also important to study any accumulation regions and potentially predict the space debris growth trends (Montebugnoli *et al.*, 2010) in near future.

In Europe, the characterisation and tracking of space debris objects is one of the primary objectives of the ESA's SST programme. Nowadays, this is done through a number of sensors in the European network of sensors designed to track these objects (Di Lizia *et al.*, 2017). In northern Scandinavia, one finds the low-frequency European Incoherent Scatter Scientific Association (EISCAT) (Klinkrad, 2006) incoherent scatter radar with a 0.70 deg beamwidth at 3 dB. Whilst the primary research goal of the facility is high-latitude atmospheric and ionospheric research, it was also shown to be effective in the detection of LEO debris of less than 2 cm at an altitude of 500 km. At present, the system is planned for an upgrade into a multi-static system (McCrea *et al.*, 2015).

The French Grand Réseau Adapté à la Veille Spatiale (GRAVES) (Klinkrad, 2002) radar system is the only space debris monitoring system outside the US Space Surveillance Network (SSN) and the Russian Space Surveillance System (SSS) which maintains an independent catalogue. Owned by the French Department of Defence (DoD), the system consists of a VHF transmitter and a planar 15 m by 6 m phased array acting as the receiver. The quoted resolution of the system, which operational tests started in 2001, is of the order of 1 m at an altitude of 1000 km. Wachtberg is home to the German Tracking and Imaging Radar (TIRA) system, of the Fraunhofer Institute for High Frequency Physics and Radar Techniques (FHR). This mono-pulse radar system consists of an L Band transmitter for tracking and a Ku band transmitter for Inverse synthetic aperture radar (ISAR) imaging (0.50° 3 dB beamwidth). The quoted detection threshold of the system is that of $\sim$2 cm at a range of 1000 km. Other European radar systems include Ukraine's Evpatoria dish antenna and the French's Armor system deployed on the Monge tracking ship. An exhaustive list of experimental and operational radar facilities for SSA is presented in (Klinkrad, 2006) and (Bonnal & McKnight, 2016).

Whilst the number of space debris monitoring sensors has grown, the implementation details of their software backends is often not readily available. This makes a direct comparison between the aforementioned sensors difficult. Consequently, this work describes the design considerations and implementation details that went into a new software backend for one of ESA's latest SSA instruments; the BIRALES bistatic radar in Italy. In so doing, it lays down the foundations on which future SSA software backends and systems can build upon. First, the components of the BIRALES project are outlined. The software backend is introduced in the section 3 whilst section 4 describes how this new software backend is used for the detection of orbital debris. Finally, a reflection on the current progress and the future objectives of this project are discussed in the concluding section.

2. The BIRALES radar

In 2014, a prototype version of the system was investigated by INAF and the Istituto di Radioastronomia (IRA) of Bologna, as part of the SSA Preparatory Program (SSA-PP). The aim of this project was to determine the suitability of Ultra High Frequency (UHF) radar measurements for space debris monitoring (Morselli *et al.*, 2015). Initial tests used the BEST-2 array in bi-static configuration with a 3 m parabolic dish transmitter located 35 km from the receiver (Morselli *et al.*, 2014). The encouraging results of these path-finding studies led to the inception of the BIRALES bi-static Space Situational Awareness (SSA) radar.

The bi-static radar makes use of a fully steerable parabolic antenna located in the Italian Joint Test Range in Caligari, Sardinia, Italy here referred to as the Radio Frequency Transmitter (TRF). The 7 m antenna has a maximum speed of $3\,^{\circ}\,s^{-1}$ and is able to supply a maximum power of 10 kW in the bandwidth 410 MHz to 415 MHz. This transmitter is designed to work both in pulsed compression mode for range measurements and in Continous Wave (CW) mode for Doppler measurements. The receiver, referred to as the BIRALES-Rx array, consists of a section of the North-South arm of the Northern Cross radio telescope located in Medicina, near Bologna, Italy (Figure 1). This gives the bistatic instrument a baseline of about 580 km.

BIRALES-Rx is a phased antenna array consisting of eight East-West oriented parabolic reflectors separated by a distance of 10 m. Each cylinder has a reflecting surface made of 430 parallel steel wires having a diameter of 0.5 mm that are placed 2 cm apart. The feed consists of 64 half-wavelength dipoles and is placed along the focal line of the main reflector. A flat sub-reflector is placed behind the focal line of the parabolic reflector in order to increase its sensitivity.

Fig. 1: The eight parabolic reflectors making up the BIRALES-Rx array within the Northern Cross, in Medicina, near Bologna, Italy

Four low noise receivers are installed on the focal line of each cylinder. Each receiver combines the dipole signals in groups of 16, resulting in four analogue channels per cylinder. This gives a total of 32 elements which are arranged in a 4 by 8 grid having a total collecting area of $1420\,m^2$ (Bolli *et al.*, 2008). When the 32-elements (8 cylinders) are used as a phased array, the signals can be used to generate the beam pattern shown in Figure 2.

The instrument has a bandwidth of around 20 MHz centred around 408 MHz, of which 16 MHz is usable. The amplified Radio Frequency (RF) signals travel to a receiver room, through 520 m long optical fibre links. These signals are down-converted to the intermediate 30 MHz frequency at the analogue block before digitisation at the receiver chain of the BIRALES radar.

The receiver chain can be split into two major sub-systems as shown in Figure 3. The first system is used to get the range of a target object whilst the second system is used to estimate the observed target's Doppler shift information and trajectory. The latter is the subject of this manuscript.

The output of both of these systems are amalgamated together at a data fusion step and saved in

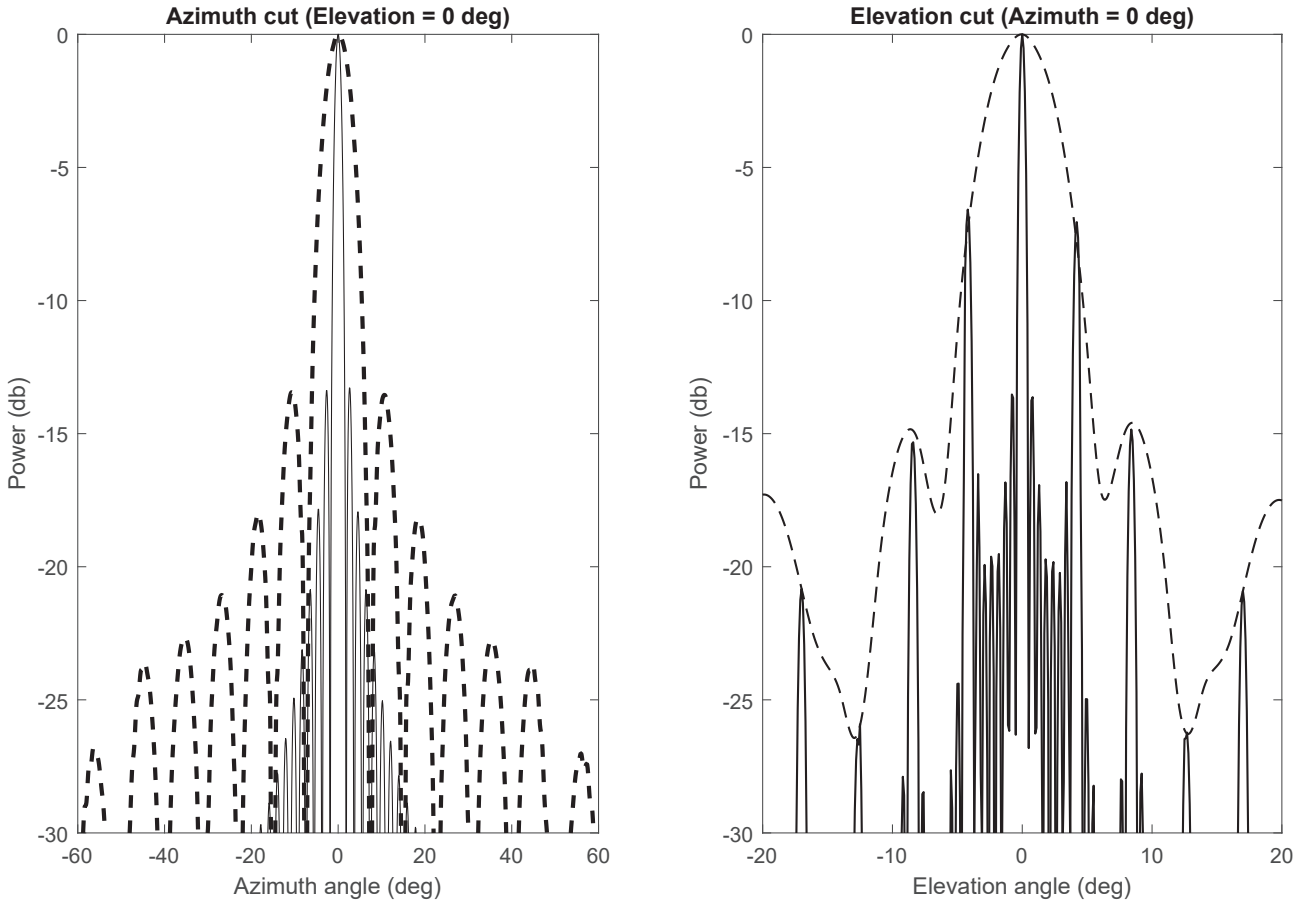

Fig. 2: The simulated synthesised beam pattern for the BIRALES-Rx phased array at 408 MHz for the end-fire case. The Azimuth cut (Elevation = 0 deg) is show on the left whilst the Elevation cut (Azimuth = 0 deg) is shown on the right. The embedded pattern for an element at the centre of the BIRALES-Rx receiver is also shown as a dashed line.

Tracking Data Message (TDM) (CCSDS, 2007) format. The TDM output serves as the input to the orbital determination block developed by researchers at the Politecnico di Milano, Italy (Losacco, 2019). This block estimates the trace of a transiting object within the receiver's field of view and refines the orbital parameters of known RSO, or performs an initial orbit determination in the case of unknown objects (Losacco, 2019).

At present, the range of a target is calculated using a 5 MHz, Frequency Modulated Continous Wave (FMCW) saw-tooth signal, that is transmitted simultaneously with the unmodulated CW carrier signal. In this configuration, the system is quoted to be capable of detecting targets at a range of 2000 km with a measurement accuracy of about 30 m. However, the BIRALES transmitter has the capability to radiate also pulsed compression signals with different settable waveforms (PRF, duty cycle, etc.). A detailed description of the BIRALES ranging system is given in (Pisanu *et al.*, 2018).

The Doppler-detection sub-system uses a Reconfigurable Open Architecture Computing Hardware (ROACH)-based digital back-end (Hickish *et al.*, 2016) adapted from (Foster *et al.*, 2014). The digital back-end digitises and channelises the signals into a total of 256 coarse frequency channels that are 78.125 kHz wide. Of the 256 channels (20 MHz) available, a single coarse channel is required in this case given that the maximum Doppler shift corresponding to an RSO is expected to be in the order of a few tens of kHz. Consequently, choosing the CW carrier frequency to be at the centre of a coarse frequency channel band

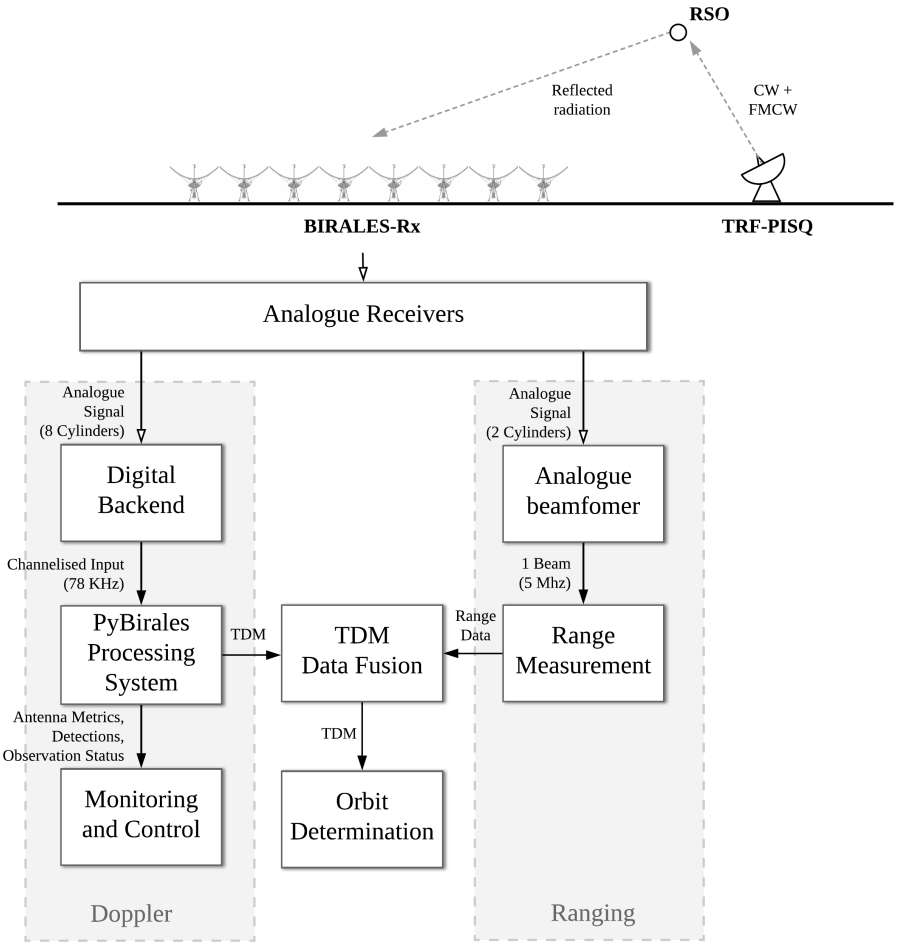

Fig. 3: A block diagram illustrating the main components of the ranging and doppler detections systems making up the BIRALES bistatic SSA radar

provides for a suitable detection window for SST. This is typically set to 410.085 MHz.

The single channel data stream is transferred as a User Datagram Protocol (UDP) stream to a processing server over a 10 Gbit link. At this processing server, the specialised data processing software developed in this work is used to analyse the incoming data stream for RSO radar signatures.

The ever-increasing computational requirements of modern radio astronomy instrumentation have led to the inception of high-performance, real-time computing installations such as those described in (Farnes *et al.*, 2018). The application of this kind of software systems can be seen in large radio instruments such as the Low Frequency Array (LOFAR) (Falcke *et al.*, 2006) and the Square Kilometre Array (SKA) (Hall, 2005).

In astronomy, a number of stream data processing libraries are now available such as Pelican (Mort *et al.*, 2015), PSR-DADA [a] and HASHPIPE [b]. More generic software such as LIGO's GStreamer library [c] can also be used, however their application to specific domains is often cumbersome. To address this, the Bifrost (Cranmer *et al.*, 2017) framework has been specifically developed to simplify the creation of new data processing software. Whilst the list of software backends in radio astronomy is long, the number of data processing systems specifically designed for RADAR application has been limited. This is particularly

[a] http://psrdada.sourceforge.net/
[b] https://github.com/david-macmahon/hashpipe
[c] https://gstreamer.freedesktop.org

true for the monitoring and tracking of uncooperative space debris objects in Near Earth Orbit (NEO). In the optical regime, there are a number of potential pipeline software as described in (Virtanen *et al.*, 2016). However, a detailed description of software systems in modern radar SST instrumentation is still limited.

As the need for new SSA instrumentation grows, so does the need for an easy-to-use, reliable, processing system that is specially built for the detection of orbital objects and can be deployed on multiple instruments. In this study, we present a new software architecture for SST. The aim is to simplify the development effort required to realise a new data processing system that is specifically designed for phased arrays. The architecture of this new software backend, named PyBirales, is described in the next section.

3. The PyBirales framework

The typical data rate of the BIRALES-Rx phased array is in the order of tens of megabytes per second. At this data rate, Python was the preferred language on which to base the new system given its development speed and availability of libraries. Whilst the data rate is relatively low when compared to those typical in radio astronomy, this new radar was designed to process the incoming data in real-time. A real-time processing of the radar data is an important prerequisite in the observation of fast transients such as orbital objects. Their detection in a timely manner is critical to time-sensitive applications such as in the observation of re-entry of orbital objects.

A high-level representation of the PyBirales architecture is illustrated in Figure 4. One can note that this three-tier architecture can be categorised into the presentation, data and application layer. The presentation tier holds the components that allow an operator to interface with and manage the system. The complexity of the system sub-processes is hidden from the operator through two user interfaces. At this layer, one finds a Command Line Interface (CLI) application and a web-based GUI that can be used to initialise, manage and monitor the software backend and instrument. One the other hand, the front-end application of PyBirales provides the user with a wealth of monitoring metrics, including system status, instrument metrics, and detection statistics emanating from the application layer.

Communication between the application and the presentation layer is perfromed through a publish-subscribe messaging pattern implemented at the data layer. PyBirales uses a REDIS [d] in-memory key-value database that acts as a messaging broker between publishers of events and their corresponding subscribers. Typically, the backend system publishes events to the message broker on a specified channel. These events are then consumed by any subscribers or listeners on that channel, such as the web application. Events can range from system warnings or errors to detection results. A single event can have multiple listeners or subscribers. This way, various aspects of the PyBirales application, such as the web application and the backend in the application layer, are completely decoupled. Decoupling the two systems is important in order to ensure that this software can be easily applied to other instruments.

The application layer contains the software components that are related to the control of the BIRALES receiver and the processing of the incoming data. At this layer, one finds a daemon service which employs a separate worker thread to listen for new events on the message broker. This worker thread interprets the control messages that are published on these channels by the front-end application. These control messages, encoded as JSON strings, can range from system kill messages to the submission of a new observation. For instance, when a new observation is created at the front-end, the application publishes the details of this observation on the designated channel. This message is received by the subscriber, in this case, the BIRALES service, which persists it to the database and adds it to the observation schedule. This schedule is a FIFO queue which hosts the pending observations. The scheduler service polls the database at frequent intervals in order to check whether the time for the next observation has elapsed. In such an instance, the observation parameters are sent to the Observation Manager which orchestrates the separate stages of an SST observation.

An observation first starts by steering mechanically the 8 parabolic cylinders of the BIRALES-Rx array towards the desired declination. Secondly, the digital backend is initialised and programmed, such that the

[d] https://redis.io

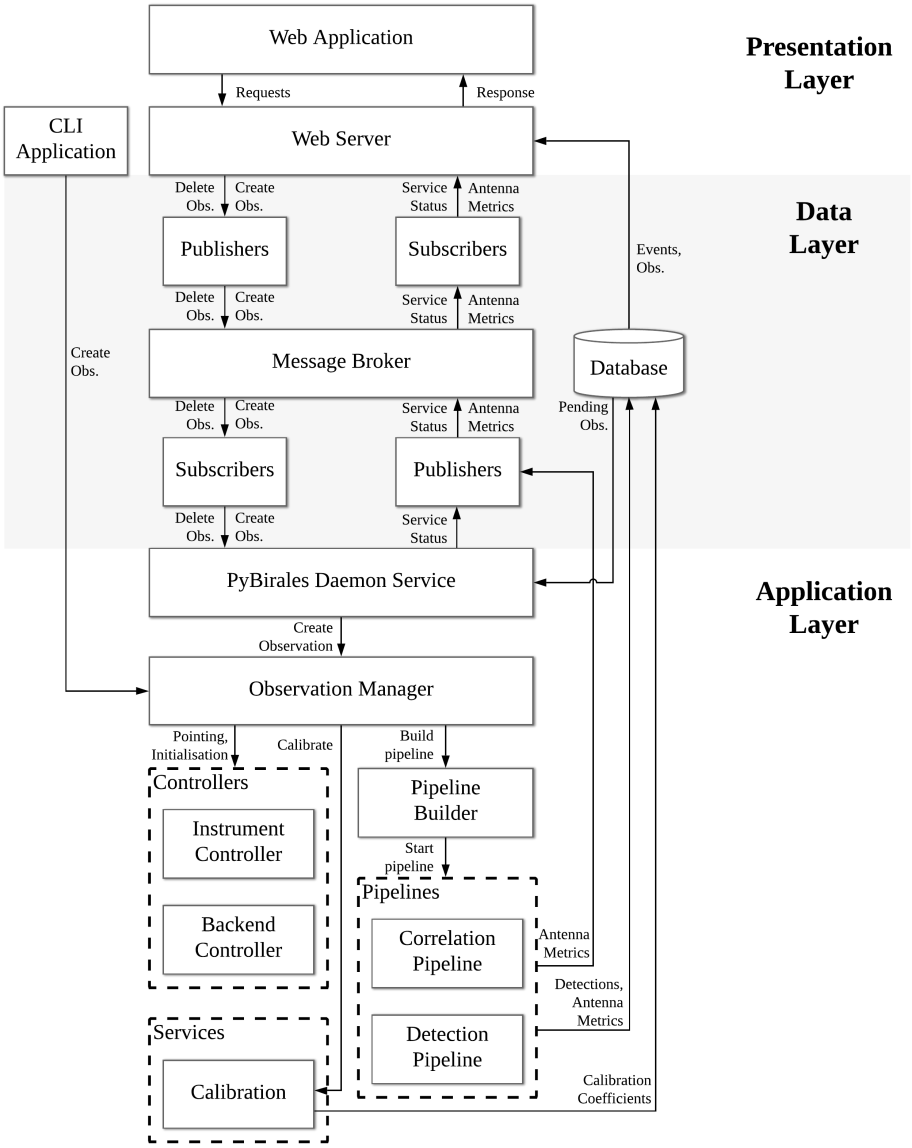

Fig. 4: An illustration of the three-tier architecture of the PyBirales space debris processing system.

antenna data starts flowing through the software backend running on a Fujitsu Celsius R940 processing server. Both the initialisation of the firmware as well as the control of the antennas is achieved through hardware controllers which manage any hardware that is external to the PyBirales system. The incoming digitised antenna signals are processed using data processing pipelines.

Software pipelines are a popular design pattern that are used in the processing of a continuous stream of data, such as in this application. A pipeline is made up of a chain of separate processing stages as shown in Figure 5. At each processing stage or module, the incoming data is mutated and passed over to the next processing stage in the chain. In PyBirales, every processing pipeline is designed to process the data in real-time. Consequently, each module in PyBirales satisfies the following condition,

$$\text{processing time} < \frac{\text{Number of samples}}{\text{Sampling rate}} \tag{1}$$

For instance, if the PyBirales is processing 2^{18} samples, at a sampling rate of 78 125 samples/s, the data should be processed in less than $\sim$3.36 s for the real-time condition to be satisfied. Pipelines are

administered by a pipeline manager which is responsible for the initialisation, processing and graceful termination of a pipeline. The construction of these pipelines can become a bit cumbersome for advanced applications. To facilitate this, and speed up the development process, PyBirales implements a framework that is able to generate a processing pipeline with ease. Using this framework, a pipeline can be assembled by chaining together a number of pre-built modules that are shipped with PyBirales.

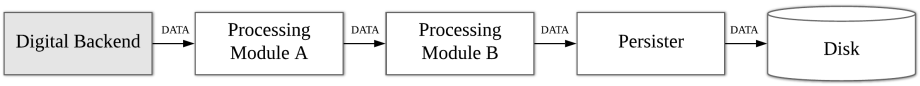

Fig. 5: A graphical representation of a linear software pipeline illustrating the processing of the data by means of a chain of modules. In this pipeline, the antenna data received from the digital backend is processed at modules A and B and persisted to the disk using a dedicated module.

A pipeline can be started through a call to the pipeline manager which initialises and starts the processing modules which is run on a separate processing thread. Once started, the data flows from one module to the next in a linear fashion. In PyBirales, a module processes what is known as a data blob and outputs a new data blob which is the input to the next processing module within the pipeline. A module can consume one particular data blob type and output another of a different type. A check on blob compatibility in-between the chained modules is performed upon the initialisation of the pipeline. Data blobs are encoded as a multi-dimensional Numpy array and act as fixed-sized circular buffers between two processing modules. These buffers can be defined by their gulp size and a buffer factor. The gulp size specifies the size of each block within the circular buffer. The buffer factor is the number of blocks that make up a circular buffer. Both of these parameters can be individually defined per module. The circular buffers make use of two pointers or indices; a read and a write pointer. A producer module writes the processed data to a block in the array specified by the write index. Upon completion, the write index is incremented to the next location. In the case where the write index reaches the end of the array, the write index is reset to the start of the array thereby overwriting the oldest block of data. Similarly, a read pointer keeps track of the last location read by a consumer module. Once a block is read by module B, the read index is incremented. As with the write pointer, the read pointer is reset to the start of the array when the index reaches the end of the array. Data can be lost if the producer module tries to overwrite a block in a location which has not been read by the consumer module. A race condition could arise given that the data blobs are being accessed by at least two module threads at any one time. One module could be trying to write to a block in an array whilst the other module is trying to read.

A deadlock is avoided by using a locking mechanism upon reading and writing. As shown in Figure 6, when a module tries to modify its output blob, which is an input blob to the next module, a write lock request is issued. When the module is finished with writing to the blob, the lock is released. A similar approach is used for reading requests. The processing module overrides the parent class' process function and implements the module specific logic there. This process function is repeated indefinitely until the processing module, together with the rest of the modules, is stopped by the pipeline manager. A processing module is stopped by flipping the stop event flag in each module.

The aforementioned data processing facilities can be used in a number of applications including the calibration of the BIRALES-Rx array. A phased array needs to be calibrated in both amplitude and phase.

The aim of the amplitude calibration is to correct the gain differences among the receiving chains, whereas phase calibration has to take into account instrumental delays. Geometrical delays arise from the fact that the plane wave-front from the source impinges on the array elements with different time delays, and such geometrical delays are well predictable and depend on both the source position and the array configuration. Conversely, instrumental delays are inherent to the instrument itself (differences in cable lengths, electronics delays, etc.) and they need to be measured by means of specific observations of bright-unresolved radio sources (calibrators).

The calibration routine is composed of a series of distinct steps as shown in Figure 7. First, the

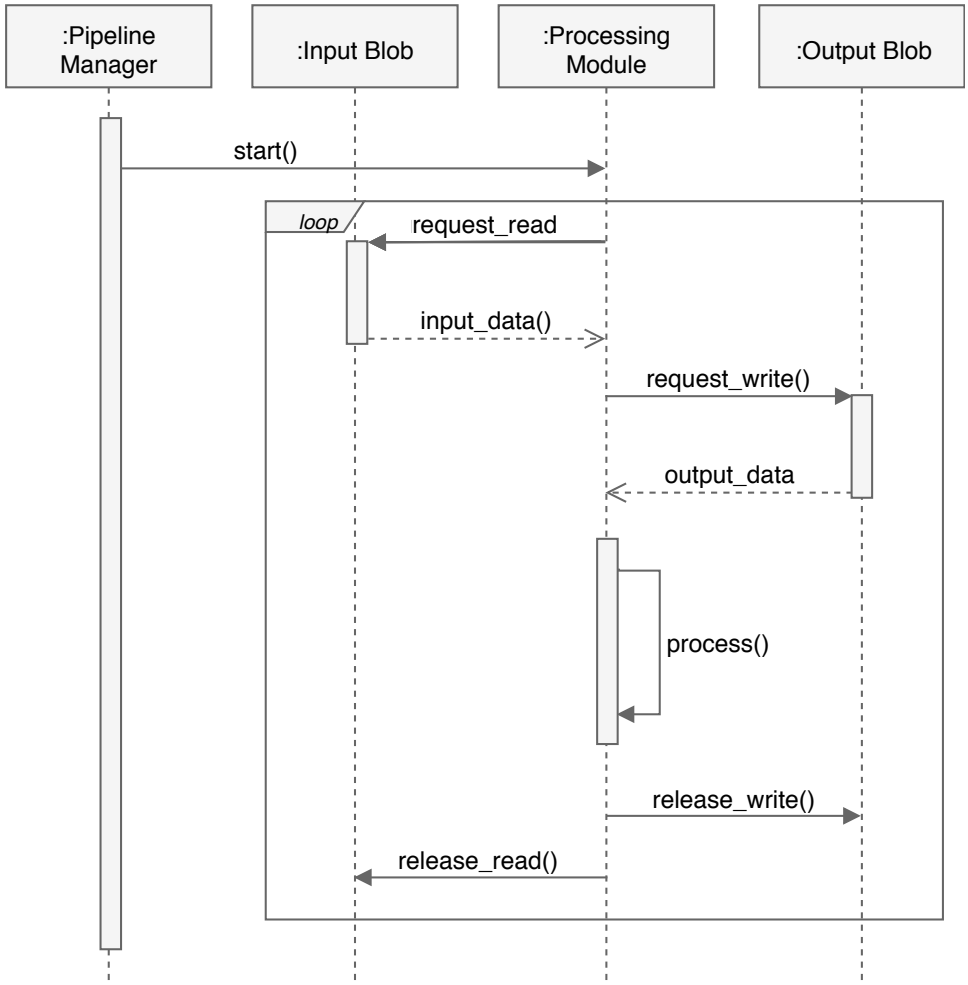

Fig. 6: A UML sequence diagram of the main process function in the processing module

correlation pipeline is used to generate the correlation matrix. This correlation matrix is populated with baseline visibilities as a calibrator radio source, such as Cassiopeia A or Taurus A, transits through the FoV of the radio telescope. The pipeline is started a few minutes before the radio source transits the local meridian, with the radio telescope pointed at the source's declination. PyBirales uses a specially-built calibration algorithm that corrects both phase and gain errors. This is achieved by using a least-squares technique to solve a system of linear equations to retrieve the unknown phase and gain coefficients.

The coefficients generated through this procedure account for the combined geometrical and instrumental delays. The geometrical delays can be calculated using the declination of the source and the antenna positions. Removing the contribution of the geometrical delays, leaves the instrumental ones, which are valid for all declinations. The generated coefficients are persisted to the database. Once the coefficients are generated, the system is said to be calibrated and the BIRALES-Rx array can be used for routine observation campaigns. As an illustrative example of this framework, the next section describes how the BIRALES radar uses the aforementioned PyBirales framework to realise a space debris detection system.

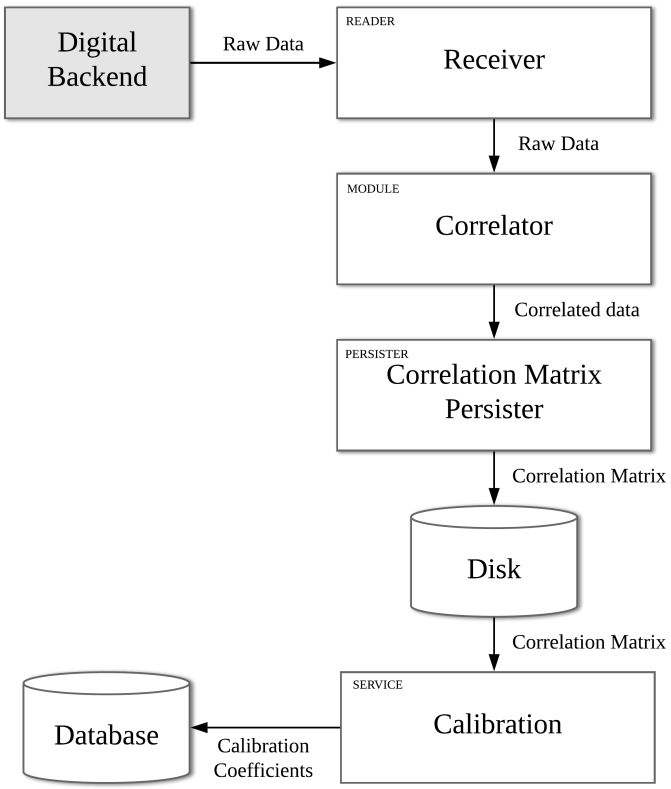

Fig. 7: Illustration of the BIRALES correlation pipeline that is used to calibrate the BIRALES-Rx phased array

4. Space debris detection pipeline

A schematic representation of the modules making up the space debris detection pipeline shown in Figure 8. The first module is the receiver module which reads the data coming from the digital backend. The format of the transmitted data is a 32-32 complex fixed-point format, which needs to be translated to floating point. In order to do so, the receiver uses the Data Acquisition (DAQ) library developed for the Antenna Verification System (AAVS) prototype of the Low-Frequency Aperture Array (LFAA) (Magro *et al.*, 2019) which was extended to accept the BIRALES data format. The receiver module reads the incoming data in chunks, converts the data into data blobs and forwards this data to the second module in the pipeline. The data generated by the receiver can also be persisted to disk. This is achieved by chaining the receiver module to a persister module that can persist the raw input data to disk as a simple binary file. This file encodes the phased array's raw data which can be subsequently processed in offline mode.

In offline mode, the pipeline uses a separate Data Reader module that reads this binary file in chunks in a similar fashion to the Receiver module. Each chunk is converted into a RawDataBlob that is forwarded to the next processing module. Hence, the system can run both in offline mode using the data reader module and in online mode using the Receiver module as the first module of the pipeline. This way, the next module in the pipeline is indifferent as to whether the pipeline is running in offline or online mode, as both reader modules output the same data blob format.

4.1. *Beamforming and multi-pixel generation*

The 32-input stream generated by the receiver module is fed to the next module of the pipeline; the beamformer. In a linear phased array consisting of N antenna elements separated by a distance d, the signals from a source with an incident angle θ, are received at a different time across the array. This time

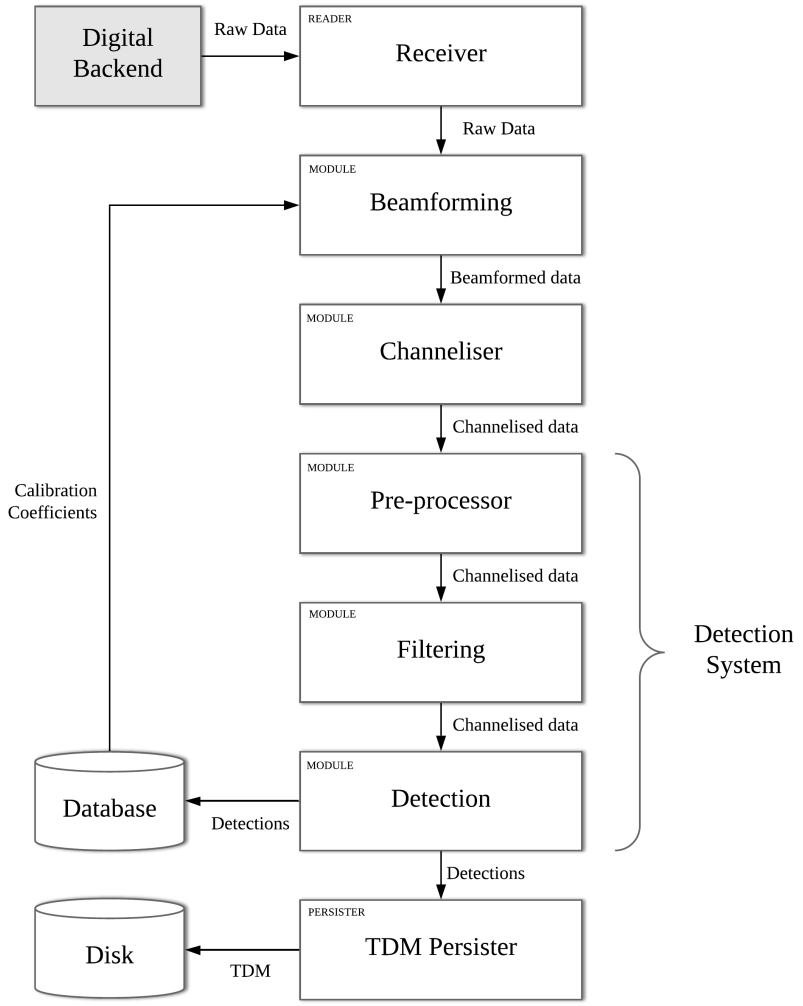

Fig. 8: Illustration of the BIRALES data processing pipeline for the detection of orbital debris

delay τ, corresponding to an antenna n within the array, is a function of the direction of propagation of the incident wave and thus will be different for all the antennas. This time delay can be corrected for by applying a phase shift (in the frequency domain) to the antenna signals such that they add constructively toward a particular direction or pointing.

Thus, the response $B(k)$ of a beam pointed towards the incident radiation with a wave-vector k_0, to radiation with other wave-vector directions $k(\phi, \theta)$, is given by:

$$B(k) = \sum_{n=0}^{N-1} A_n(k) \exp\left(i(k - k_0) \cdot r_n\right) \tag{2}$$

where $A_n(k)$ is the response of an individual antenna, r_n is the position vector of the antenna n. Additionally, the point at which the signals add constructively can be changed by varying the phase shift applied to the antennas signals. Thus, a beam can be steered to point in any direction within the instrument's FoV.

Applying this principle, multiple beams, each with a different pointing direction can be created from the same input by copying the input signal and applying a different phase shift. In so doing, the instrument's FoV can be covered by multiple beams pointed at different directions simultaneously. The generated beams cover the FoV of the antenna into what is henceforth referred to as the multi-pixel.

In PyBirales, the generation and steering of these beams is done at the Beamformer module. The number of beams generated, together with their pointing, is configurable and can be specified by the operator of the PyBirales software backend. At present, the default configuration generates 32 coherent beams within are electronically steered inside the instrument's 5.7° by 6.6° FoV. Figure 9, illustrates the arrangement of these beams at −3 dB.

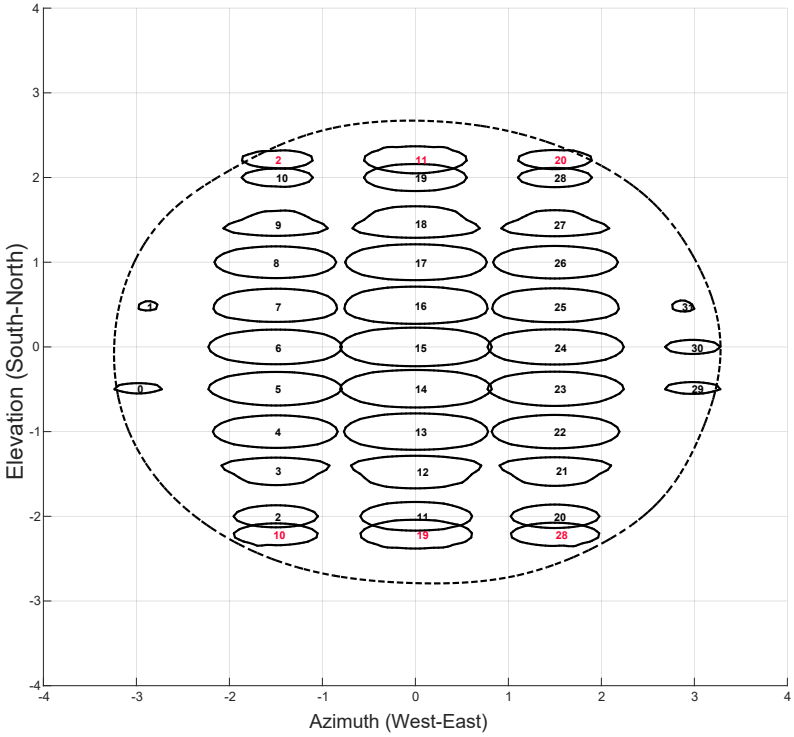

Fig. 9: The multi-pixel configuration of the BIRALES radar. The large blue ellipse shows the single element pattern at −3 dB. The beams that are affected by ambiguity due to grating lobes are highlighted in red.

One may observe that each major lobe of this configuration is narrow, however grating lobes, marked in red, start to appear at an offset of 4.2° in the H-Plane (for the end-fire case). The maxima corresponding to grating lobes are not desirable in radar applications. Grating lobes introduce ambiguities in the detection and can hinder the radar's ability to distinguish between a detection in the main lobe with that inside of a grating lobe. Grating lobes are unavoidable in the BIRALES-Rx array given that the separation distance d between the antenna elements (along the South-North direction) is greater than $\lambda/2$ (for the end-fire case).

Figure 10 shows the normalised power of Cassiopeia A as it transits across the four central beams (6, 15, 24, 30 in Figure 9) generated by the PyBirales beamformer. One may also observe the aforementioned aliasing effect which is shown to become more severe as the beam is steered away from the array phase centre.

The operator can generate any number of beams within the instrument's FoV. However, the computational expense grows with the number of beams generated. In fact, the complexity of the module is $\mathcal{O}(N_c N_b N_s N_a)$, where N_b is the number of beams, N_c is the number of channels, N_s is the number of samples and N_a is the number of antennas. The Beamformer's main processing routine is a Python wrapper to a C++ beamforming implementation. At the last stage, the Beamformer module applies the calibration coefficients which correct for instrumental errors, as described earlier.

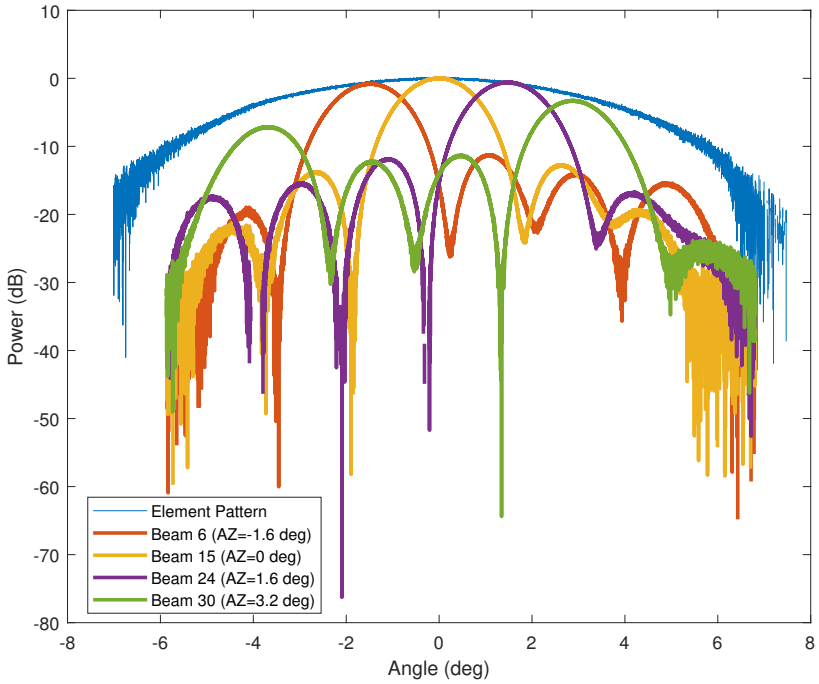

Fig. 10: The normalised power observed in four beams pointed at a declination of 58.9 deg during a transit of Cassiopeia A.

4.2. *Channelisation*

The frequency bandwidth of a channel cannot be greater than the expected maximum Doppler shift. Thus, the 78.125 kHz needs to be channelised further such that the spatial resolution is increased. Typically, a channel can be split into finer channels by a Discrete Fourier Transform. However, Fast Fourier transform (FFT) channelisation can result in leakages into boundary frequency channels. To counter this, a Polyphase Filter Bank (PFB) channelizer with N taps was used since spectral leakage is significantly lower.

The PFB channelizer implemented within the Channeliser module splits the coarse channel into 8192 separate channels of ∼9.5 Hz each. This gives a temporal resolution of ∼ 10 samples per second. A radar signature can be detected on multiple channels within the same time epoch. This phenomenon, known as doppler migration, manifests itself in radar streaks with a high SNR. This causes the reflected radiation to be detected in one channel to leak into the adjacent channels as a consequence of the FFT channelisation being used. The PFB channeliser minimises this effect when compared to a standard FFT channelizer however any Doppler migration detected is handled by considering only the channel with the highest SNR per time epoch. This is usually the one at the centre of the Doppler spread.

Channelising the data is computationally expensive. The computational complexity of the Channeliser is $\mathcal{O}(N_c N_a(log_2(N_c) + 8N_t))$ N_b is the number of beams, N_c is the number of channels, N_t is the number of taps. This complexity necessitated the use of the Python Numba library (Lam *et al.*, 2015). Numba speeds up the processing of the application whilst maintaining the application syntactic eloquence. Numba is used to compile native python code to native machine instructions. In so doing, Numba-annotated parts of a Python application can achieve a level of performance similar to that achieved by their C or C++ equivalent.

The objective of the project is to gradually extend the number of antennas used in the radar receiver array. As the array grows, so does the computational requirements of the system. The performance of the modules is dependent on the size of the input data of the module. Figure 11 shows how the current implementation of the Channeliser and Beamformer modules is able to process up to four times the current

data rate within the real-time constraint. In the case of the Beamformer, this is equivalent to generating up to 128 independent beams. Similarly, quadrupling the existing input of the Channeliser is the equivalent of channelising the input of 32 cylinders. One can observe that all the modules are well below the real-time threshold of 3.355 s per module.

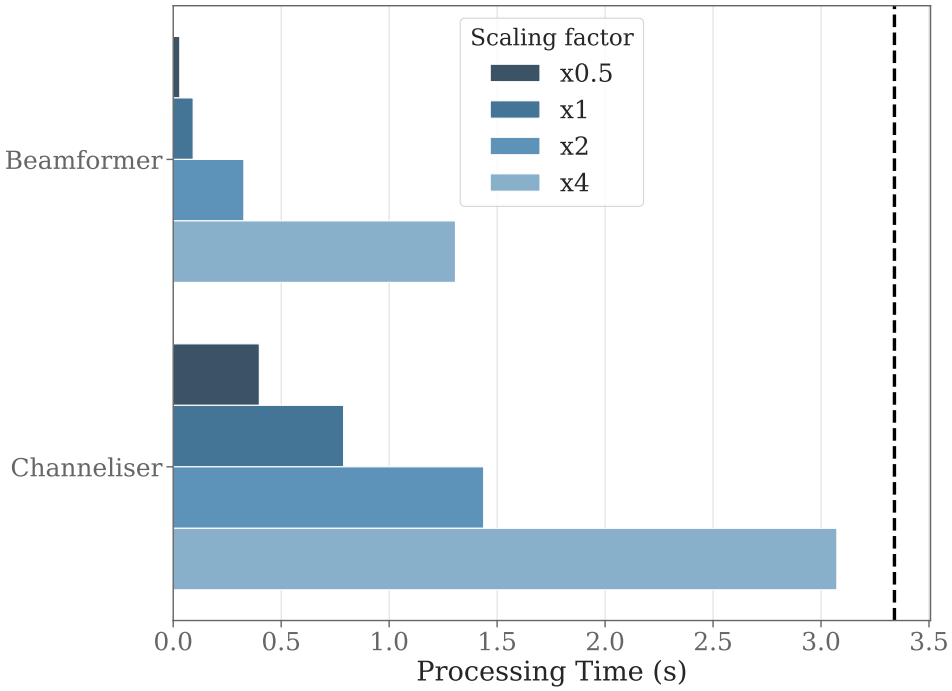

Fig. 11: Scaling of the Channeliser and Beamformer module in PyBirales. Dotted line represents the real-time criteria. The benchmarks were run on the production workstation having two Intel Xeon E5-2640 processors and 32 GB of DDR3 memory

Figure 12, illustrates the typical channelised data blob of the PyBirales system of a single beam after channelisation. As one can note, this data is essentially a spectrograph of N channels by M samples for each beam produced by the Beamformer module. The output captures the linear radar streak that is typical of a high-velocity orbiting objects. In a strong-enough detection, side-lobe detections can also be observed. This is the pattern of interest to be detected in the subsequent modules within the detection pipeline.

4.3. *Detection*

The channelised data is passed through a series of filters that remove the background noise and Radio-Frequency Interference (RFI) signals as much as possible without reducing the quality of the target streaks. Subsequently, the Detector module, detailed in (Cutajar *et al.*, 2018), uses a hierarchical clustering approach to identify high-intensity radar streaks within the image.

The streaks identified by the detection algorithm are validated and saved as a measurement set consisting of the detection's epoch, Doppler shift and SNR. These measurements are used to infer the time at which the target passes over each beam. As the target passes over the FoV of the instrument, it is detected across a number of beams within the multi-pixel at different times depending on its trajectory. The order in which the different beams are illuminated by the target objects radar echo is known as the beam illumination sequence.

The measurements are exported in TDM format and transferred to a remote server to be used in the first phase of the orbital determination algorithm. The algorithm makes use of the SNR profile of the detections together with the beam illumination sequence to infer the trajectory of the object inside the receivers FoV. This estimate is further refined by combining this data with range information obtained by

a separate ranging system that is installed on-site. A detailed description of the trace reconstruction and state estimation phases can be found in (Losacco, 2019).

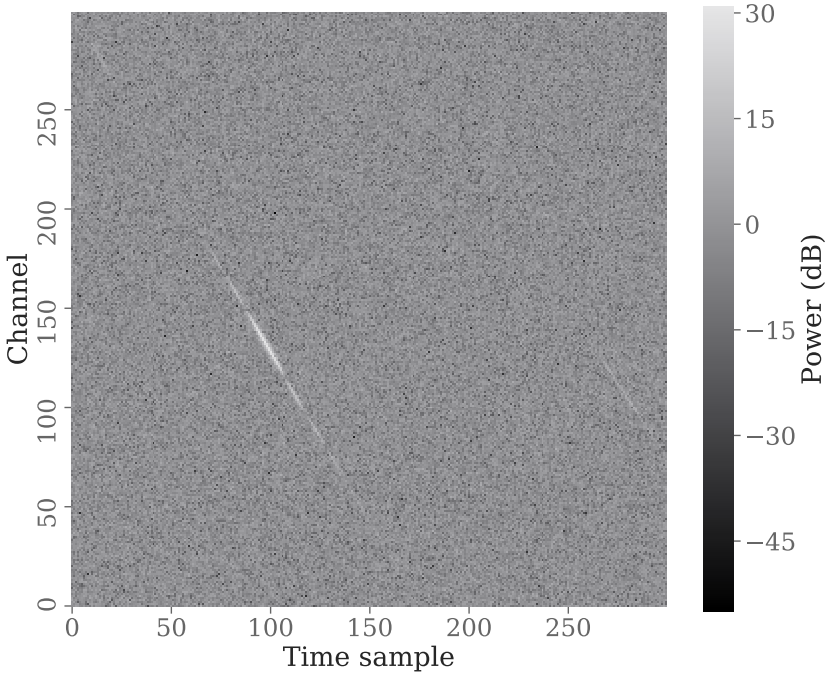

Fig. 12: A subset of the channeliser output for a single beam that illustrates the radar signature of a target RSO in the presence of background noise. NORAD 25160 detection on February 11^{th}, 2019

PyBirales saw its first operational use in early 2018 with the re-entry of the Tiangong-1 space station. The beam illumination sequence, showing the maximum SNR per time epoch across all beams, is illustrated in Figure 13. It can be noted that the radar echo activates 14 beams of the multi pixel. A strong SNR was measured in each of these beams, with the highest SNR value of 46.35 dB being measured in the central beam 15. Assuming the transit time of the RSO is the time at which the highest SNR was measured, the target object crossed beam 15 on the 29^{th} of March 2018 at 07:56:01.558 UTC at a Doppler shift of -4511.795 Hz. These results were in agreement with the expected values. In fact, the Two-Line Element (TLE) estimate for the object days before its re-entry was expected to be -4425.9636 Hz at 07:56:15.775 UTC. Since then, the BIRALES radar has been involved in a number of tests and operational campaigns. Targeted campaigns have shown a capability of detection of objects with an RADAR Cross Section (RCS) of 0.11 m^2 at a slant range of up to 1800 km (Cutajar *et al.*, 2018).

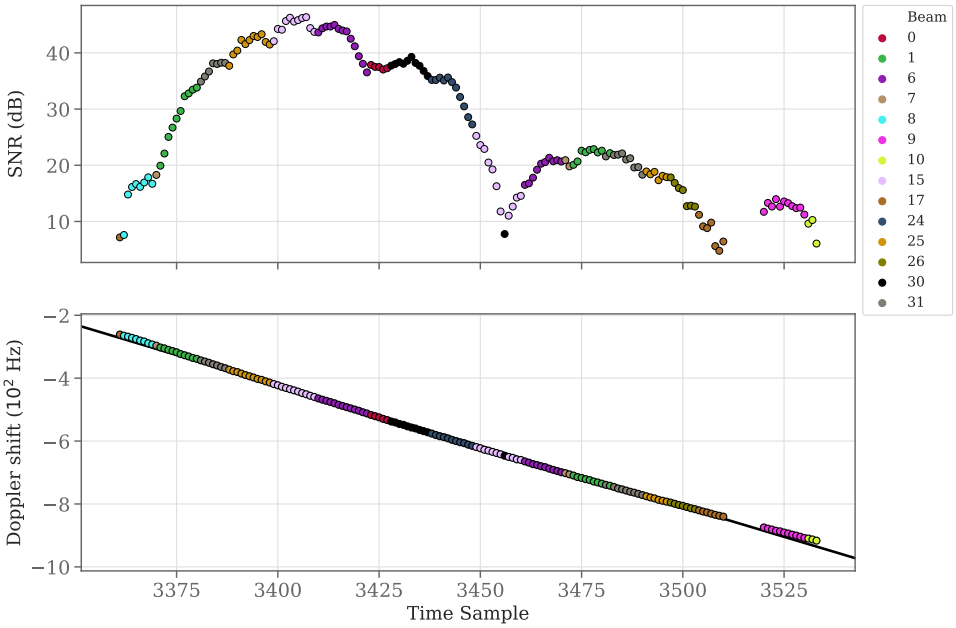

Fig. 13: The SNR profile across the activated beams (top). The measured Doppler shift against time sample (bottom). NORAD 37820 (Tiangong-1) detection on March 29^{th}, 2018.

5. Conclusion

As the population of orbital debris increases, so does humanity's responsibility towards Earth's space environment. An international research effort is needed to monitor the space environment using all the available national assets. The realisation of a new SSA radar situated in Medicina necessitated the implementation of a new software backend. The goal of this software backend was to enable the researchers to control the BIRALES-Rx array and process the data from its 32 receivers using an innovative and highly configurable system.

An operator can use the PyBirales framework to create data processing pipelines that process one 78 kHz wide channel from a ROACH-based digital backend. In this manuscript, it was shown how a highly configurable, real-time processing system can be implemented with ease by making use of a chain of processing modules. These modules modify the incoming data before forwarding them to the next module in the chain. This highly configurable architecture made it possible to use the same system for a variety of applications, including the calibration of the array and the detection of orbital debris. Consequently, it was shown how a radio telescope that is typically used for radio astronomy application can be easily adapted for use in SSA application.

The system is operational and has already shown its capability in the monitoring of catalogued objects. At present, it is being used in routine beam park experiments for unknown objects. Whilst these results are promising, new functionality to the system are constantly being implemented. Potentially, the processing capability of the system could be extended to cater for an additional number of receivers. It is predicted that this upgrade may require some of the modules, such as the detector, to be optimised to account for the increased data rate. The ultimate goal of these advances is to build an accurate and reliable orbital debris monitoring system that meets the European overarching SSA commitment.

Acknowledgements

The research activities and operations described in this paper were developed thanks to the funding from the European Commission Framework Programme H2020 and Copernicus, SST Space Surveillance and Tracking contract No. 785257-2-3SST2016 and No. 237/G/GRO/COPE/16/8935-1SST2016. The Italian Air Force's TRF transmitter, is located at Italian Joint Test Range of Salto di Quirra in Sardinia, Italy. The Northern Cross Radio Telescope is a facility of the University of Bologna operated under agreement

by the IRA-INAF (Radio Astronomy Institute National Institute of Astrophysics).

References

Bolli, P., Perini, F., Montebugnoli, S., Pelosi, G. & Poppi, S. [2008] *IEEE Antennas and Propagation Magazine* **50**, 58.

Bonnal, C. & McKnight, D. S. [2016] "IAA Situation Report on Space Debris," Tech. rep., International Academy of Astronautics (IAA).

CCSDS [2007] "Recommendation for Space Data System Standards TRACKING DATA MESSAGE RECOMMENDED STANDARD," Tech. rep., The Consultative Committee for Space Data Systems (CCSDS), URL https://public.ccsds.org/Pubs/503x0b1c1.pdf.

Cranmer, M. D., Barsdell, B. R., Price, D. C., Dowell, J., Garsden, H., Dike, V., Eftekhari, T., Hegedus, A. M., Malins, J., Obenberger, K. S. *et al.* [2017] *Journal of Astronomical Instrumentation* **6**, 1750007.

Cutajar, D., Magro, A., Borg, J., Zarb Adami, K., Bianchi, G., Bortolotti, C., Cattani, A., Fiocchi, F., Lama, L., Maccaferri, A. *et al.* [2018] "A real-time space debris detection system for birales," *69th International Astronautical Congress (IAC 2018)*, p. 1.

Di Lizia, P., Massari, M., Losacco, M., Bianchi, G., Mattana, A., Pupillo, G., Bortolotti, C., Roma, M., Morselli, A., Armellin, R., Magro, A., Cutajar, D., Portelli, C. & Reali, M. [2017] "Performance assessment of the multibeam radar sensor birales for space surveillance and tracking," *7th European Conference on Space Debris*, p. 1.

Falcke, H. D., van Haarlem, M. P., de Bruyn, A. G., Braun, R., Röttgering, H. J., Stappers, B., Boland, W. H., Butcher, H. R., de Geus, E. J., Koopmans, L. V. *et al.* [2006] *Proceedings of the International Astronomical Union* **2**, 386.

Farnes, J., Mort, B., Dulwich, F., Salvini, S. & Armour, W. [2018] *Galaxies* **6**, 120.

Foster, G., Hickish, J., Magro, A., Price, D. & Zarb Adami, K. [2014] *Monthly Notices of the Royal Astronomical Society* **439**, 3180, doi:10.1093/mnras/stu188.

Grassi, M., Cetin, E. & Dempster, A. G. [2015] *IEEE Transactions on Aerospace and Electronic Systems* **51**, 1231, doi:10.1109/TAES.2015.140129.

Hall, P. J. [2005] *The square kilometre array: An engineering perspective*, Vol. 2 (Springer).

Hickish, J., Abdurashidova, Z., Ali, Z., Buch, K. D., Chaudhari, S. C., Chen, H., Dexter, M., Domagalski, R. S., Ford, J., Foster, G. *et al.* [2016] *Journal of Astronomical Instrumentation* **5**, 1641001.

Klinkrad, H. [2002] *International Colloquium on Europe and Space Debris* .

Klinkrad, H. [2006] *Space debris* (Springer), p. 5, ISBN 354025448X, doi:10.1002/9780470686652.eae325.

Lam, S. K., Pitrou, A. & Seibert, S. [2015] "Numba: A llvm-based python jit compiler," *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (ACM), p. 7.

Losacco, M. e. a. [2019] "The multibeam radar sensor BIRALES: performance assessment for space surveillance and tracking," *Proceedings of the 40th IEEE Aerospace Conference* (Montana, USA), p. 1.

Magro, A., Bugeja, K., Chiello, R. & DeMarco, A. [2019] "A high-performance, flexible data acquisition library for radio instruments," *2019 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, p. 069, doi:10.1109/APWC.2019.8870490.

McCrea, I., Aikio, A., Alfonsi, L., Belova, E., Buchert, S., Clilverd, M., Engler, N., Gustavsson, B., Heinselman, C., Kero, J., Kosch, M., Lamy, H., Leyser, T., Ogawa, Y., Oksavik, K., Pellinen-Wannberg, A., Pitout, F., Rapp, M., Stanislawska, I. & Vierinen, J. [2015] *The science case for the EISCAT_3D radar*, Vol. 2 (Progress in Earth and Planetary Science), ISBN 4064501500, doi:10.1186/s40645-015-0051-8, URL http://dx.doi.org/10.1186/s40645-015-0051-8.

Montebugnoli, S., Pupillo, G., Salerno, E., Pluchino, S. & di Martino, M. [2010] *Advances in Space Research* **45**, 676, doi:10.1016/j.asr.2009.09.011, URL http://dx.doi.org/10.1016/j.asr.2009.09.011.

Morselli, A., Armellin, R., Di Lizia, P., Bernelli-Zazzera, F., Salerno, E., Bianchi, G., Montebugnoli, S., Magro, A. & Adami, K. Z. [2014] "Orbit Determination of Space Debris Using a Bi-Static Radar Configuration with a Multiple-Beam Receiver," ISBN 9781634399869, p. 1.

Morselli, A., Lizia, P. D., Bianchi, G., Bortolotti, C., Montebugnoli, S., Naldi, G., Perini, F., Pupillo, G., Roma, M., Schiaffino, M., Mattana, A., Salerno, E., Magro, A., Adami, K. Z., Armellin, R., Sergiusti, A. L., Villadei, W., Dolce, F., Reali, M. & Paoli, J. [2015] "A new high sensitivity radar sensor for space debris detection and accurate orbit determination," *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, p. 562, doi:10.1109/MetroAeroSpace.2015.7180719.

Mort, B., Dulwich, F., Williams, C. & Salvini, S. [2015] "Pelican: Pipeline for Extensible, Lightweight Imaging and CAlibratioN," .

Pisanu, T., Schirru, L., Urru, E., Ortu, P., Inaf, C., Scienza, V., Ca, S., Bianchi, G., Bortolotti, C., Roma, M., Nazionale, I., Ira, R. I., Fiorentina, V., Bo, M., Muntoni, G., Montisci, G., Protopapa, F., Podda, A., Sulis, A. & Valente, G. [2018] *2018 22nd International Microwave and Radar Conference (MIKON)* , 317.

Virtanen, J., Poikonen, J., Säntti, T., Komulainen, T., Torppa, J., Granvik, M., Muinonen, K., Pentikäinen, H., Martikainen, J., Näränen, J., Lehti, J. & Flohrer, T. [2016] *Advances in Space Research* **57**, 1607, doi:10.1016/j.asr.2015.09.024.