



An interference detection strategy for Apertif based on AOFlagger 3

A. R. Offringa^{1,2} , B. Adebahr³ , A. Kutkin¹, E. A. K. Adams^{1,2}, T. A. Oosterloo^{1,2}, J. M. van der Hulst², H. Dénes¹, C. G. Bassa¹, D. L. Lucero^{4,1}, W. J. G. Blok^{1,5,2}, K. M. Hess^{6,1,2}, J. van Leeuwen¹, G. M. Loose¹, Y. Maan^{7,1}, L. C. Oostrum^{1,8,9}, E. Orrú¹, D. Vohl^{8,1}, and J. Ziemke^{1,10}

¹ ASTRON, the Netherlands Institute for Radio Astronomy, Oude Hoogeveensedijk 4,7991 PD Dwingeloo, The Netherlands
e-mail: offringa@astron.nl

² Kapteyn Astronomical Institute, University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands

³ Astronomisches Institut der Ruhr-Universität Bochum (AIRUB), Universitätsstrasse 150, 44780 Bochum, Germany

⁴ Department of Physics, Virginia Polytechnic Institute and State University, 50 West Campus Drive, Blacksburg, VA 24061, USA

⁵ Dept. of Astronomy, Univ. of Cape Town, Private Bag X3, Rondebosch 7701, South Africa

⁶ Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía s/n, 18008 Granada, Spain

⁷ National Centre for Radio Astrophysics, Tata Institute of Fundamental Research, Pune 411007, Maharashtra, India

⁸ Anton Pannekoek Institute, University of Amsterdam, Postbus 94249, 1090 GE Amsterdam, The Netherlands

⁹ Netherlands eScience Center, Science Park 402, 1098 XH Amsterdam, The Netherlands

¹⁰ University of Oslo Center for Information Technology, PO Box 1059, 0316 Oslo, Norway

Received 20 September 2022 / Accepted 3 January 2023

ABSTRACT

Context. Apertif is a multi-beam receiver system for the Westerbork Synthesis Radio Telescope that operates at 1.1–1.5 GHz, which overlaps with various radio services, resulting in contamination of astronomical signals with radio-frequency interference (RFI).

Aims. We analyse approaches to mitigate Apertif interference and design an automated detection procedure for its imaging mode. Using this approach, we present long-term RFI detection results of over 300 Apertif observations.

Methods. Our approach is based on the AOFlagger detection approach. We introduce several new features, including ways to deal with ranges of invalid data (e.g. caused by shadowing) in both the SumThreshold and scale-invariant rank operator steps; pre-calibration bandpass calibration; auto-correlation flagging; and HI flagging avoidance. These methods have been implemented in a new framework that uses the Lua language for scripting, which is new in AOFlagger version 3.

Results. Our approach removes RFI fully automatically, and it is robust and effective enough for further calibration and (continuum) imaging of these data. The analysis of 304 observations shows an average of 11.1% of lost data due to RFI with a large spread. We observe 14.6% RFI in auto-correlations. Computationally, AOFlagger achieves a throughput of 370 MB/s on a single computing node. Compared to published machine learning results, the method is one to two orders of magnitude faster.

Key words. instrumentation: interferometers – methods: observational – techniques: interferometric – surveys – radio continuum; general

1. Introduction

The technical advancement of mankind is driving an increase in man-made radio-frequency transmitters, both terrestrial and in space. This increases the challenge for radio astronomical studies that try to detect sky signals that are many orders of magnitude fainter than man-made transmissions. Now that radio-astronomy is evolving into a science where it is the norm to measure data volumes in petabytes, mitigation of radio-frequency interference (RFI) needs to be computationally efficient and fully automated.

Apertif is a receiver system upgrade for the Westerbork Synthesis Radio Telescope (WSRT) that makes use of phased-array feeds to allow for 40 simultaneous adjacent beams on the sky (Van Cappellen et al. 2022). Observations are performed at a central frequency of 1280 or 1370 MHz with an instantaneous bandwidth of 300 MHz.

The data volume produced by Apertif is considerable. Voltages from the 12 dishes with Apertif receivers are correlated for all beams, typically integrated for 30 s and recorded with four polarizations. The bandwidth of 300 MHz is split into

384 sub-bands, each with 64 channels of 12.2 kHz. Because of the large bandwidth, it overlaps with various services, including GPS and air-traffic communications. Although the WSRT resides in a radio protected zone, it is not shielded from satellites or air-traffic. Moreover, starting in 2020, 5G transmissions have made use of the 1452–1492 MHz bandwidth. For these reasons, Apertif requires an efficient approach to deal with RFI. Due to the large amount of data, such an approach has to work fully automatically.

The most common method to deal with RFI is to detect data samples that have a significant contribution of RFI and ignore affected data in the processing (e.g. Winkel et al. 2006; Middelberg 2006; Offringa et al. 2010a; Prasad & Chengalur 2012; Peck & Fenech 2013; Yang et al. 2020; Sun et al. 2022). This process is referred to as data flagging, and is also our method of choice for dealing with RFI in Apertif data in this work. Our detection methodology builds upon the RFI detection pipelines for the Low-Frequency Array (LOFAR; Van Haarlem et al. 2013; Offringa et al. 2010b) and the Murchison Wide-field Array (MWA; Tingay et al. 2013; Offringa et al. 2015).

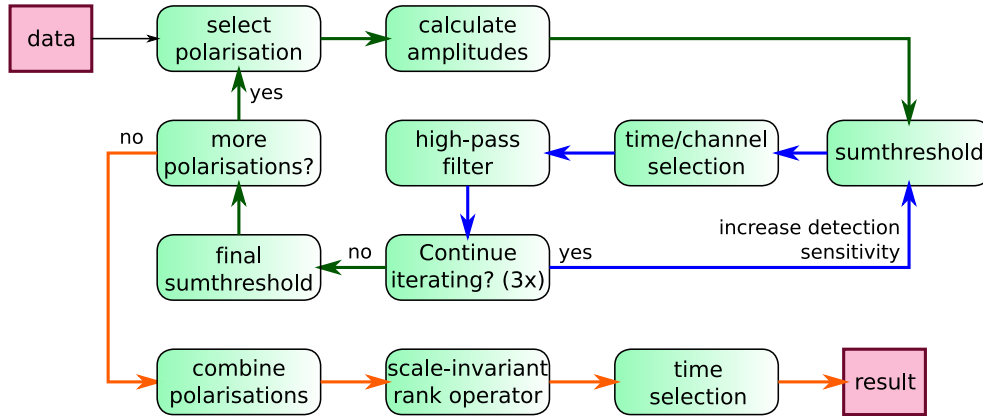


Fig. 1. Default AOFLAGGER strategy for RFI detection (before modifications for Apertif). These steps were independently performed on smaller subsets of the data. The input data of one independent run through these steps typically consists of approximately an hour of correlations from a single pair of antennas and a single beam, with the full bandwidth and all four linearly polarized cross correlations present.

Those pipelines integrate an AOFLAGGER flagging strategy, which combines filtering, sumthresholding, morphological operations, and heuristics. Details of the AOFLAGGER approach are discussed in Sect. 2.1.

Apertif supports a transient (beam-formed) mode and an imaging mode. The RFI detection approach for these two modes are fundamentally different. In this work we aim at an RFI detection in imaging mode, that is, after having correlated and integrated the voltages from all the antennas. Readers can refer to Sclocco et al. (2019) for an approach to mitigate RFI in beam-forming mode. Our approach is part of a fully automated Apertif imaging pipeline called APERCAL (Adebahr et al. 2022).

A multi-beam receiver makes it possible to perform spatial filtering techniques to suppress interference (Kocz et al. 2010, 2012; Hellbourg et al. 2014). This requires fast dedicated computing hardware that processes the raw signals from all the beams, which for Apertif is not available. Spatial filtering is also mainly used to filter out a limited number of known transmitters, which for Apertif is likely not sufficient by itself, although it might save some part of the bandwidth.

Another approach to detect interference is by using the spectral kurtosis statistic (Gary et al. 2010; Taylor et al. 2019; Purver et al. 2021). This has shown results that are competitive with amplitude-based detection. However, this requires a specialized correlator and doubling the data volume to be able to calculate the kurtosis.

Recently, machine learning has been used to address the issue of RFI detection (Harrison & Mishra 2019; Yang et al. 2020; Xiao et al. 2022; Sun et al. 2022). Yang et al. (2020) argue that convolutional neural networks can achieve an accuracy that is higher than that of their SUMTHRESHOLD implementation. For this comparison, the authors use their own customized implementation of the SUMTHRESHOLD method, whereas in platforms such as AOFLAGGER the method is typically applied iteratively and combined with filters (Offringa et al. 2010a,b) and morphological operators (Offringa et al. 2012; Van de Gronde et al. 2016) to enhance the accuracy. With these additions, it has been shown that pipelines such as AOFLAGGER typically detect all interference that astronomers would manually flag. In this work, we showcase what can be achieved with traditional methods – including their computational requirements – thereby providing an updated baseline to compare against.

In this paper, we introduce a flagging strategy for Apertif data using the AOFLAGGER framework, and demonstrate our

designed strategy on Apertif data. In Sect. 2, we start by introducing the AOFLAGGER steps used to construct the Apertif approach, and introduce several new operations that are integrated into the Apertif flagging strategy. In Sect. 3, results of applying this strategy are presented, including long-term statistics and the computational requirements. Finally, in Sect. 4 we discuss the results and draw conclusions.

2. Method

For this work, we have designed an interference detection approach for Apertif based on the existing AOFLAGGER approach and integrated this into the APERCAL pipeline. APERCAL is an automated processing pipeline for Apertif imaging observations (Adebahr et al. 2022), consisting of common steps such as data formatting, interference detection, calibration and imaging. Interference detection is one of the first steps during data reduction and is fundamental for achieving a good and persistent calibration and image quality and later steps of the processing.

To improve the detection quality, several modifications to AOFLAGGER are required. This consists of extensions of existing algorithms and optimizing parameters for APERCAL, which we subsequently discuss in this section. We start with an overview of the detection approach.

2.1. Overview

Figure 1 shows an overview of the steps that the default AOFLAGGER strategy performs. The AOFLAGGER approach to RFI detection in a subset can be summarized as (i) estimation and subtraction of the sky signal by applying a Gaussian high-pass filter in time-frequency space (see Sect. 2.5); and (ii) detection of excessive values, with increased sensitivity towards spectral-lines and broadband features. The detection is performed with the SUMTHRESHOLD algorithm (Offringa et al. 2010a). Steps i) and ii) are typically iterated three times with increased sensitivity to make sure that the final sky signal estimate is minimally biased by interference. As a final step, the flags from different polarizations are combined and are extended in time and frequency, using the scale-invariant rank (SIR) operator (Offringa et al. 2012; Van de Gronde et al. 2016). This latter step improves detection of interference that tapers off below the noise floor and

fills gaps in the flag mask when a persistent transmitter is not fully detected.

With AOFLAGGER, detection of interference is performed independently on subsets of the data, and the pipeline of Fig. 1 runs independently for each subset. For Apertif, such a subset was chosen to contain the data from all four linearly polarized correlations (XX, XY, YX, YY), the full bandwidth (300 MHz), an interval of typically half an hour for a single beam and a single correlated baseline. Hence, the detection of interference for different beams, baselines and time intervals is independently performed, even though these are part of the same observation. The motivation for flagging these subsets independently is two fold: (i) It improves performance, as it allows parallel and distributed detection of subsets. The independent flagging of beams and time intervals matches with the format of the data. Despite this, data access is still not ideal, because the data for one baseline is stored dispersed over the time direction; (ii) Combined detection does not significantly improve detection, as the added value of detection on combined subsets of data is small, that is, one subset contains little information about the RFI in another subset. This is because the impact of RFI can vary greatly between different beams and different baselines. Furthermore, it rarely occurs that RFI which affects image quality is not detectable in half an hour of data, but is detectable when multiple half hour intervals are combined.

Performing detection on integrated baselines has, in some cases, been shown to make faint RFI detectable (Offringa et al. 2015; Wilensky et al. 2019). Early tests with Apertif data, however, indicated that there is no gain in combining baselines. We have also performed tests that flag after integrating over multiple beams, but again found no improvement in doing so. These tests were not exhaustive and it could be that combined detection on baselines or beams could still improve the accuracy somewhat.

AOFLAGGER aims to take out RFI that requires raw, high-resolution data flagging. Because of the high resolution of processed data, the computational performance of detection is critical. It is important to perform high-resolution flagging early, because it results in the highest accuracy and the impact of flagging is reduced compared to low-resolution flagging (Offringa et al. 2013). On the other hand, some phenomena cause the loss of large time intervals or frequency ranges. Common instrumental causes are correlator failures, temporary local RFI, or strong broadband transmitters. Detection of such issues does not require the high-resolution data, and it is therefore less critical to detect such issues in the first AOFLAGGER detection run. Such issues can be found in post-processing of lower-resolution data for which the performance is less critical.

2.2. Invalid data

There are several instrumental issues that may result in data with invalid values that interrupt the data in time or frequency. A few examples of such issues are correlator malfunctions, dish shadowing, incorrectly set sub-band gains, network failures (between stations and the correlator) or data corruption. Such instrumental issues result in visibilities that may have non-physical values for certain times, frequencies, feeds or antennas, or could lead to visibilities with a not-a-number (NaN) value. We refer to such data as invalid data.

In most cases, invalid data can be detected and flagged early in the processing. For example, shadowing can be determined from the target direction and the layout of the array, and missing sub-band data caused by network congestion can be detected by the correlator. In this paper, we consider the detection of such

issues outside the context of interference detection. It does, however, make it necessary for the detector to continue to work in the presence of (pre-detected) invalid data, which may affect only specific times, frequencies or some other selection of data.

Making the AOFLAGGER algorithm aware of invalid data is one of the changes that was required for Apertif. The AOFLAGGER algorithm was originally designed to work on raw high-resolution single-subband LOFAR data. It rarely happens that such a span of data is partially invalid, and initially AOFLAGGER algorithms therefore do not take invalid data into account. In the case of Apertif, the full bandwidth is offered to AOFLAGGER, and the loss or corruption of a single subband causes therefore gaps in the bandwidth. Being a different instrument, Apertif is also affected by different issues that may not affect LOFAR, such as shadowing. For these reasons, we have extended the AOFLAGGER algorithm to take invalid data into account. This requires changes to the SUMTHRESHOLD and SIR-OPERATOR steps of the algorithm, which we subsequently discuss in the next two sections.

2.3. Extension of the SUMTHRESHOLD algorithm

The SUMTHRESHOLD algorithm is a combinatorial thresholding method that detects line-like structures in the time-frequency data (Offringa et al. 2010a). This method is effective for the detection of RFI, because most RFI raises the amplitude of consecutive time or frequency samples. The method iteratively thresholds the average over an increasing number of neighbouring samples with a decreasing threshold. With i the zero-indexed iteration number, M_i the number of samples, χ_i the threshold and ρ a constant normally chosen to be 1.5,

$$M_i = 2^i \quad (1)$$

$$\chi_i = \chi_0 \rho^{-\log_2 M_i} \quad (2)$$

We note that χ_0 is a user parameter that controls the total sensitivity of the method. The various default AOFLAGGER algorithms use values of $\chi_0 = 6 \dots 8.5\sigma$. The mode of the noise σ is determined from the data that is (at that point in the detection) determined to be RFI free, and is estimated by calculating the truncated mode of the RFI free data, skipping 20% of the outlier values (the 10% minimum and maximum values), thereby assuming that the inner 80% follow a Rayleigh distribution. Assuming that the contribution of the noise is Gaussian distributed in the real and imaginary components of the visibilities, this results in a stable estimate of its standard deviation (Fridman 2008).

A single iteration consists of thresholding all sequences of size M_i in both the time and the frequency direction (unless $M_i = 1$), possibly with different thresholds for the two dimensions, to separately control the sensitivity towards spectral line RFI and transient broadband RFI. Typically, a total of 9 of these iterations are performed, giving a maximum size of $M_8 = 256$. A sample that is flagged in an earlier iteration or direction, is (temporarily) replaced by the mean of the non-flagged samples in the sequence. The following description demonstrates the first three iterations, using $\chi_0 = 6$ and $\rho = 1.5$:

1. Flag samples with an absolute value $\geq 6\sigma$.
2. (a) Flag every sequence of 2 consecutive samples in time with an absolute average $\geq 4\sigma$ (because $\chi_2 = 6\sigma \times 1.5^{-\log_2(2^2)} = 4\sigma$).
- (b) Flag every sequence of 2 consecutive samples in frequency with an absolute average $\geq 4\sigma$.

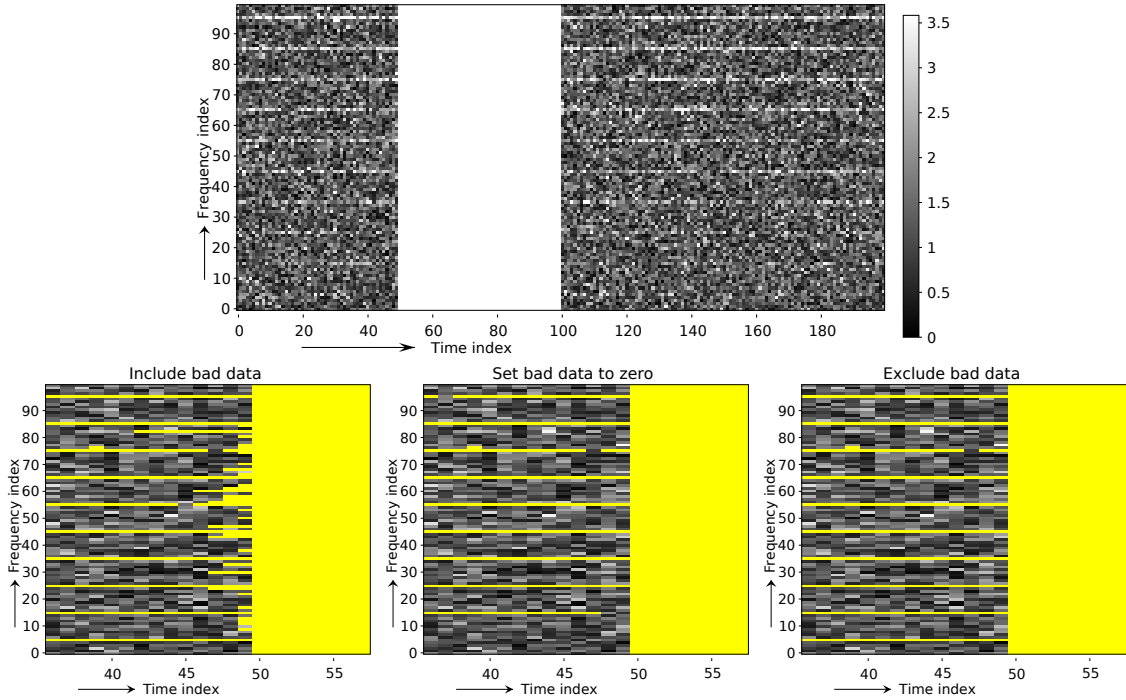


Fig. 2. Three methods of handling invalid data in the SUMTHRESHOLD step. The top image shows the simulated input data, which consists of Gaussian complex noise, spectral line RFI every 10 channels that increases in strength in frequency direction, and a block of invalid data (time indices 50–100), simulating e.g. a temporary correlator failure. The bottom images show a zoom in on the left edge of the invalid data. Flagged data is marked in yellow. Bottom-left: normal SUMTHRESHOLD without using knowledge of the invalid data; bottom-centre: invalid samples are set to zero before SUMTHRESHOLD; bottom-right: invalid samples are removed before SUMTHRESHOLD.

3. Repeat 2.(a) and (b) with 4 samples and a threshold of $\chi_4 = 6\sigma \times 1.5^{-\log_2(2^3)} = 2\frac{2}{3}\sigma$.

Subsequent iterations will threshold sequences of 8, 16, 32, ... samples with an average above $\chi_8 \approx 1.8\sigma$, $\chi_{16} \approx 1.2\sigma$, etc.

In the form described by Offringa et al. (2010a), pre-existing classification of invalid data is not taken into account in the SUMTHRESHOLD method. An example of such a case is shown in Fig. 2, which considers a simulated observation with 200 timesteps and 100 channels. The observation contains spectral-line interference that affects one channel out of every ten channels and increases power at higher frequencies. Timesteps 50–100 are known to be invalid data, and are set to high values by raising them with 10 times the standard deviation.

The second row of Fig. 2 zooms in on time indices 30–60. The first image of the second row shows the result of a basic application of SUMTHRESHOLD. For this result, the knowledge that some data was invalid is not used. As a result, the invalid data is considered to be RFI, and samples before and after the block of invalid data are flagged with an increased sensitivity. As a result, the false-positive rate is clearly increased.

A simple approach to mitigate this is to consider invalid values to be zero when applying the SUMTHRESHOLD method. This results in the plot shown in the middle of the second row of Fig. 2. This result does not show increased false positives because of the invalid data. With this approach, information about flagged samples on either side (before or after) of the missing data does not (significantly) aid detection, because the invalid data is considered to be zero, and this lowers the average absolute sum in the iterations of the SUMTHRESHOLD method that consider longer consecutive ranges. This results in a higher false-negative rate than would theoretically be possible if the information on both sides of the invalid data would have been

used together. In particular, the faintest interfering line at channel index 5 is no longer detected.

While the loss in accuracy is minimal, there is a simple method to aid the detection of interference on one side of the block of invalid data with information from the other block: by completely skipping data in the summed direction (time or frequency). In other words, samples that are directly before and after a block of invalid data are treated as if they are consecutive. The result of this is shown in the third column of Fig. 2, which indeed shows a lower false-negative rate. In particular, the faintest spectral line at channel 5 is now fully detected.

When comparing these two approaches to deal with invalid data, the approach to exclude the invalid data leads to a small increase in false-positive detections when the RFI is not consistently present in time or frequency, that is, when it is present on one side of the invalid data block and not present on the other side. This should be weighted against the increased sensitivity when the RFI is consistently present. The optimal choice therefore depends on the behaviour of the RFI. Because persistent RFI is common, and because it is more important to avoid false negatives in persistent RFI (which might negatively affect later processing steps) over avoiding false negatives in transient RFI (which would lead to a small loss of data), we use the method of excluding invalid data in our Apertif strategy.

We have implemented this in two ways: (i) stack all valid data into a temporary storage, run the normal SUMTHRESHOLD algorithm on these data and reverse the stacking operation on the resulting mask; and (ii) skip over the invalid data inside the SUMTHRESHOLD method. We have timed these two implementations on simulated complex Gaussian data with 10 000 timesteps \times 256 channels. Each run is repeated 100 times. The first implementation runs about $2.5\times$ faster (0.18 s per data

set) compared to the second implementation (0.45 s per data set). The first method is still 6× slower compared to the regular algorithm (which takes 0.03 s per data set). This can be explained by the extra copying of data that is required in each iteration (both for the time and for the frequency direction).

2.4. Extension of the SCALE-INVARIANT RANK OPERATOR

The SIR-operator is a morphological operation that is used in AOFLAGGER to extend the detected RFI mask in the time and frequency direction. It is an effective step to follow threshold-based methods to detect faint RFI based on the morphology of detected flags (Offringa et al. 2012; Van de Gronde et al. 2016). It is scale invariant, which implies that the fractional increase in flags in one dimension is constant, that is, independent of the scale of that feature in that dimension.

The SIR-operator is essentially a one-dimensional operator that can be applied to a sequence of flag values. To apply it to radio interferometric data, Offringa et al. (2012) applied the operator in both the time and frequency dimensions: in time it is separately applied to all the channels, and in frequency it is applied separately to all timesteps. The union of these two steps is taken as the result.

We define X as a single sequence of flag values, such that $X[i]$ holds a Boolean value that represents the state of the flag. The output $\rho(X)$ of the SIR-operator applied to X , is defined as the union of all subsequences of the input X , for which

$$\#_{\mathcal{F}}^{i:j} \geq (1 - \eta)(j - i). \quad (3)$$

Here, $\#_{\mathcal{F}}^{i:j}$ is brief for $\#_{\mathcal{F}}(X[i : j])$, which is the count-operator that returns the number of flagged samples in a sequence. $X[i : j]$ is the subsequence of samples consisting of all elements $X[k]$ for $i \leq k < j$ and $\eta \in [0 \dots 1]$ is a tunable parameter that sets the aggressiveness of the operator.

Equation (3) implies that a sequence of flags caused by invalid data is extended on both sides by a ratio of η . An example of this is given in the centre-left panel of Fig. 3. This behaviour is undesirable because, unlike most RFI signals, invalid data typically has a sharp boundary and should be flagged like that. The extension of invalid data causes a high number of false positives.

A simple solution is to count invalid data as unflagged data in the SIR operator. This implies that Eq. (3) is modified so that the count operator only counts the number of flags corresponding to valid data:

$$\#_{\mathcal{F}\mathcal{V}}^{i:j} \geq (1 - \eta)(j - i), \quad (4)$$

where $\#_{\mathcal{F}\mathcal{V}}$ is the number of valid samples that are flagged in the interval $X[i : j]$ (as opposed to $\#_{\mathcal{F}}$, which counts flagged values that can both be valid or invalid). Because the right side is unchanged and the left side remains equal or is decreased compared to Eq. (3), this modification always flags an equal or fewer amount of samples. An application of this approach is demonstrated in the centre-right panel of Fig. 3. This approach remedies the extending of flags around invalid data.

The downside of the approach of Eq. (5) is that a continuous transmitter is assumed not to be present in the invalid data range, causing flags on either side to have a decreased probability of getting flagged. For example, in case a correlator fails for a minute during which a transmitter remains present in one channel with decreasing power, the transmitter is less likely to be flagged after the correlator failure. To address this, we further modify Eq. (3) to:

$$\#_{\mathcal{F}}^{i:j} \geq (1 - \eta) \#_{\mathcal{V}}^{i:j}, \quad (5)$$

where $\#_{\mathcal{V}}^{i:j}$ is the number of valid (flagged or unflagged) samples in interval $X[i : j]$. This approach is effectively the same as removing the invalid samples from the sequence before applying Eq. (3). Therefore, a transmitter that gets interrupted by invalid data receives a higher probability to get flagged. An example of this approach is given in the bottom-left panel of Fig. 3. Invalid samples are skipped in this approach, and flagged samples on one side of a sequence of invalid samples may increase the probability of samples on the other side of the sequence, irregardless of the size of the invalid sample sequence.

The approach of Eq. (5) can overstep its goal of using information from before and after a sequence of invalid data, in particular in the case of very long sequences of invalid samples. For example, when considering a transmitter that is active for one minute before the receiving antenna is shadowed for 6 h (causing invalid data), it is undesirable that samples after shadowing receive higher detection probability because of what happened 6 h ago. A final modification to the SIR operator we consider is therefore to introduce a penalty parameter ρ that can balance between Eqs. (4) and (5):

$$\#_{\mathcal{F}}^{i:j} \geq (1 - \eta)((j - i)\rho + \#_{\mathcal{V}}^{i:j}(1 - \rho)). \quad (6)$$

With $\rho = 0$, invalid samples are skipped, making the method equal to Eq. (5) and with $\rho = 1$, invalid samples are counted as unflagged samples, making the method equal to Eq. (4). A value of $\rho = 0.2$ implies that five invalid samples count as one unflagged sample, thereby lowering the probability of flagging through a block of invalid data, but still transferring some of the flag information from before to after the invalid data and vice versa. This method is demonstrated with a setting of $\rho = 0.1$ in the bottom-right panel of Fig. 3.

Considering the results of all approaches in Fig. 3, it is clearly undesirable to generally extend invalid data using the traditional SIR-operator defined in Eq. (3). Any of the three different variations of the algorithm (Eqs. (4)–(6)), which can be described by choosing different ρ -values in Eq. (6), solve this issue. The different values of ρ do not cause significant changes. We have tested values of ρ on a few observations, some with artificially added invalid data, and visually compared the flagging results. Based on these results and the arguments given earlier about finding a balance between Eqs. (4) and (5), we use $\rho = 0.1$.

Introducing the parameter for invalid-data weighting ρ has no significant effect on the speed of the algorithm. The original algorithm can be implemented with a computational complexity of $O(N)$ (Offringa et al. 2012), and the same holds for the algorithm that includes the invalid-data penalty parameter.

2.5. High-pass filtering

The high-pass filter that is applied to remove astronomical source contribution before thresholding is, for computational reasons, implemented as a Gaussian low-pass filter followed by subtracting the difference between the input and the low-pass filtered result. The high frequency resolution of Apertif makes it necessary to use a large filtering kernel in the frequency direction. Effectively, a kernel with a Gaussian sigma of 875 channels and 2.5 timesteps is used. Before filtering, the data is averaged in the frequency direction by a factor of 175, and after low-pass filtering, the result is upsampled to the original resolution using nearest neighbour resampling. This allows a convolution with a much smaller kernel, improving the speed of this operation. The result is an approximate of a Gaussian high-pass filter, but for the purpose of removing the sky signal, this is sufficiently accurate.

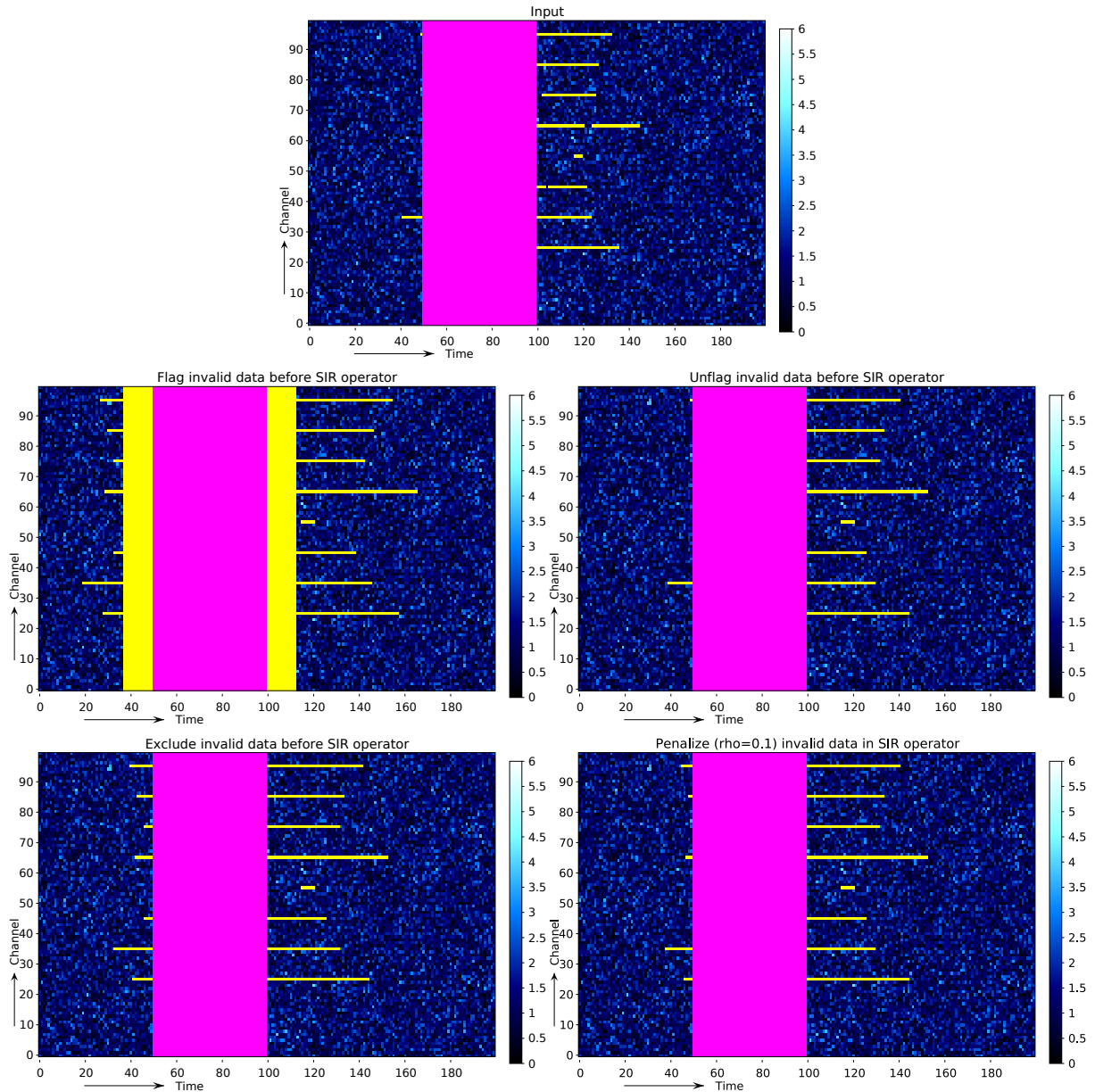


Fig. 3. Different ways of handling invalid data in the SIR-OPERATOR step on a simulated data set with a Gaussian burst of interference in a few channels. Purple marks invalid data, yellow is detected as interference. The SIR-operator operates on the flag mask, hence the visibility values are not used. Top: input data. Centre-left: invalid data is counted as flagged data. Centre-right: Invalid data is counted as unflagged data. Bottom-left: invalid data is removed before applying the SIR-operator. Bottom-right: Invalid data is penalized with $\rho = 0.1$.

2.6. Bandpass correction

In the Apercal Apertif processing pipeline, the entire bandwidth of Apertif is used at once during RFI detection. This is different from the original LOFAR strategy, that flagged small (200 KHz) subbands independently. Using the entire bandwidth has the benefit that broadband RFI that covers several sub-bands can be detected. This is relevant for Apertif observations, which are affected by broadband transmitting satellites and radar.

Because the bandwidth of Apertif is subdivided into subbands using a poly-phase filter bank, the band shape of the poly-phase filter is imprinted on the data. An example of this is shown in the top-left panel of Fig. 4. This is corrected for during calibration, but during flagging (which needs to be done before calibration) the shape is still present.

Performing detection using the entire bandwidth but without correcting for the poly-phase filter bank causes sub-band edge channels to be flagged, because the edges cause sharp transitions that trigger the detector. Moreover, the deviations in the data caused by the band-edges decrease the sensitivity of the detection towards actual RFI. The top-right panel of Fig. 4 shows an example of flagging without bandpass correction.

To remedy this, we implement a sub-band band-pass correction step in the detector. This step corrects the poly-phase filter shape using a static, observation-independent correction. We determine the shape by performing gain-calibration on a clean region of the band, and average the solutions over the subbands. The bottom-left panel of Fig. 4 shows the resulting corrected data set, and the bottom-right panel of Fig. 4 shows the result of flagging the bandpass. As can be seen, the band-pass

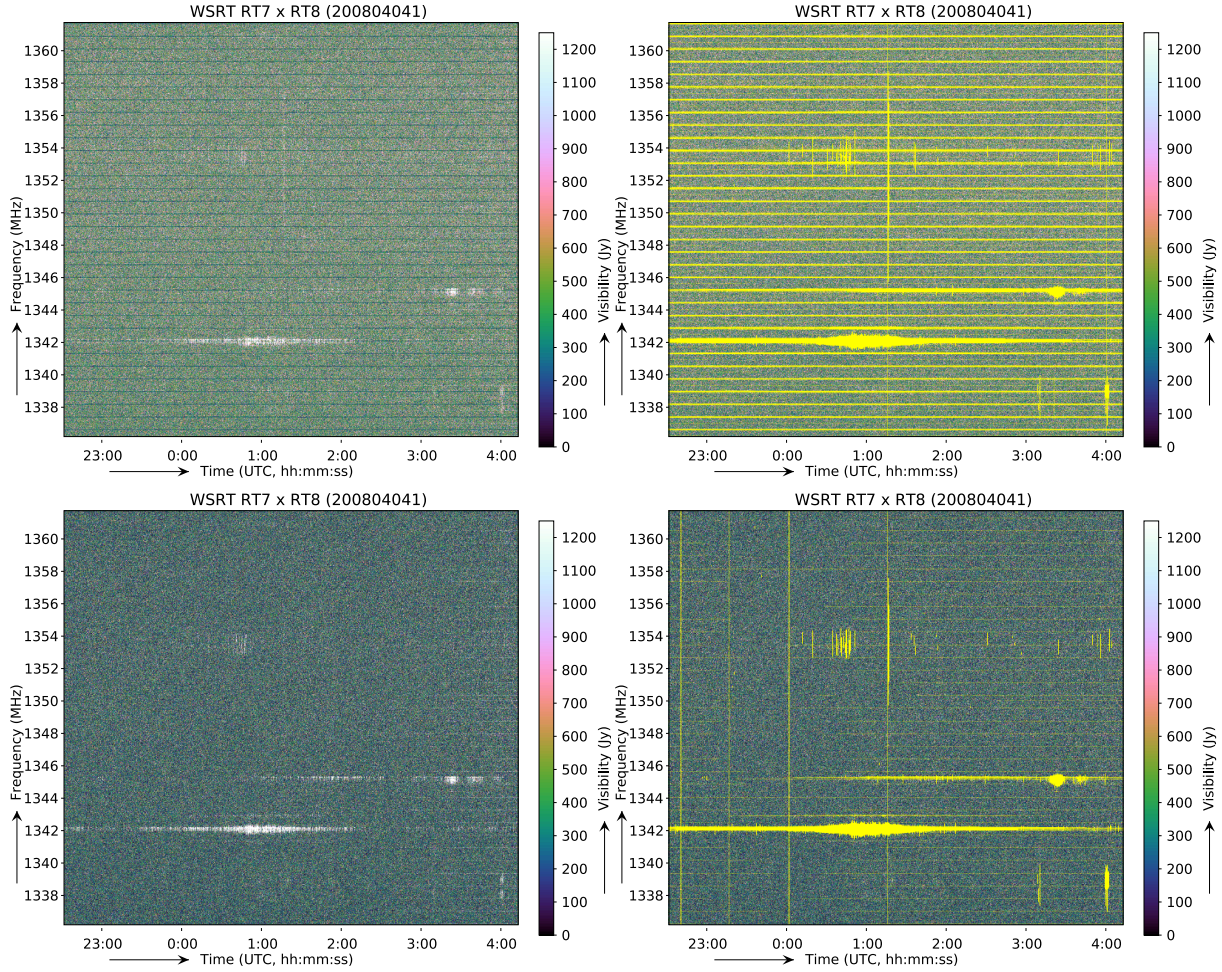


Fig. 4. Static sub-band band-pass correction before flagging with the Apertif flagging strategy. Top-left: input before correction; top-right: flagged without correction; bottom-left: input after correction; bottom-right: flagged with correction.

correction has decreased the number of false detections considerably. Some edge channels are still flagged, even after correction. This is caused by aliasing in the sub-band edge channels, which change the statistics of those edge channels slightly. This can lead to artefacts which are very similar to RFI, hence they are occasionally flagged. This flagging is normally of limited concern, because those sub-band edge channels that are flagged are of lower quality. Because of this, they are often discarded during imaging.

2.7. Flagging of auto-correlations

Given the output voltage of the two feeds of the same antenna, $\mathbf{e} = (e_x, e_y)$, auto-correlated visibilities are formed by taking the product $\mathbf{e}^H \mathbf{e}$ (i.e. the outer product $\mathbf{e} \otimes \mathbf{e}$) and integrating, resulting in XX , XY , YX and YY visibilities. While auto-correlations are not often used for scientific data products, they are useful for system monitoring and quantifying the system noise. For such analyses, it is desirable to flag RFI.

Compared to cross-correlated visibilities, auto-correlated visibilities have different properties: in the XX and YY correlations, system noise and RFI will not decorrelate, and auto-correlated visibilities are sensitive to the global sky signal instead of fluctuations in the sky signal. An example of auto-correlated dynamic spectrum from Apertif is shown in the top image of Fig. 5 (after sub-band band-pass correction as

described in Sect. 2.6). Compared to cross-correlations such as shown in Fig. 4, the dynamic spectrum of auto-correlated visibilities appears much smoother, is systematically offset from zero and contains stronger structure in the frequency direction.

The flagging strategy that was optimized for the cross-correlations detects RFI by comparing high-passed filtered amplitudes of visibilities to the variance of these amplitudes. Because the amplitude variance is much lower compared to cross-correlations, this results in flagging auto-correlations with increased sensitivity. At the same time, the auto-correlations contain stronger instrumental frequency-structure. These two effects combined causes the cross-correlation flagging strategy to flag all of the visibilities of the auto-correlations of Fig. 5.

To solve this, we use a different flagging configuration for the auto-correlations. The difference with the cross-correlation strategy is as follows: (i) The time-direction $SUMTHRESHOLD$ step (sensitive to consistently high values in the time direction, e.g. band-pass structure) is reduced in sensitivity by a factor of 6; (ii) The frequency-direction $SUMTHRESHOLD$ step (sensitive to consistently high values in the frequency direction, e.g. broadband RFI) is reduced in sensitivity by a factor of 2; (iii) The size of the high-pass filter kernel is reduced by 3.5 in the frequency direction, to filter out more of the spectral gain fluctuations of the instrument; (iv) The number of iterations is increased from 3 to 5. This increases the required computations but improves robustness in the presence of a large dynamic range, as is the case for

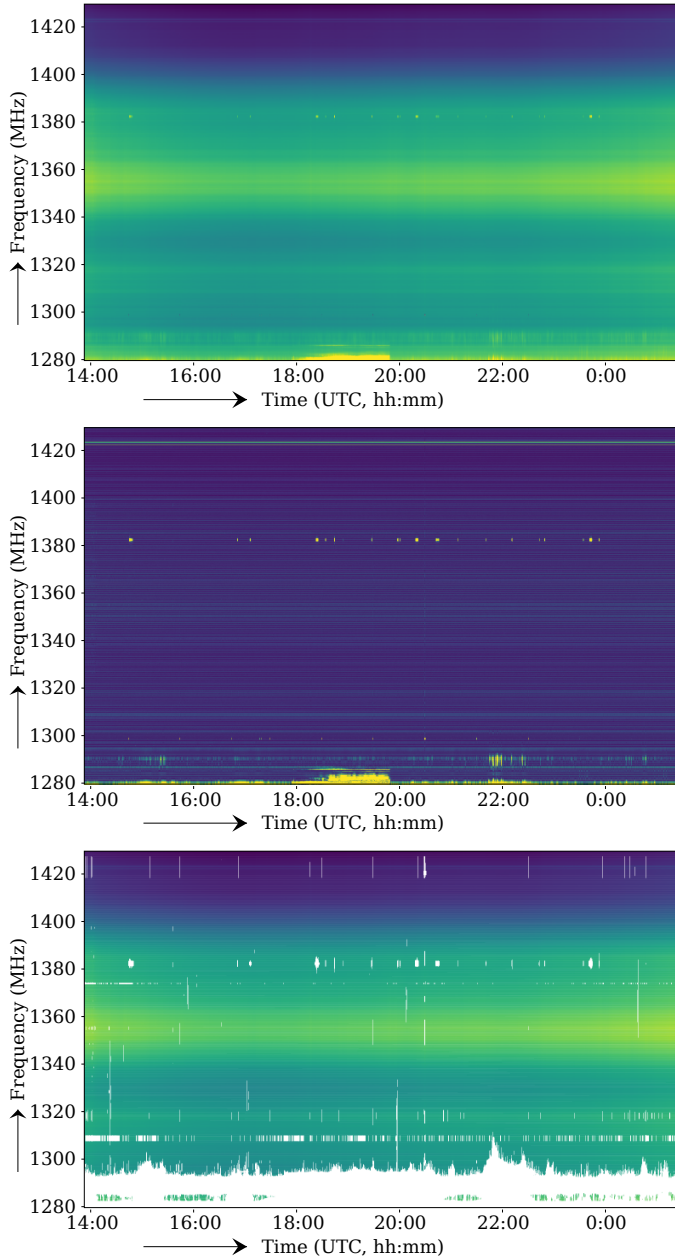


Fig. 5. Flagging of auto-correlations. Top image: input after sub-band band-pass correction; centre image: same after iterative high-pass filtering and with 10x more sensitive colour scale; bottom image: after flagging with the auto-correlation specific strategy. Because auto-correlations have different properties compared to cross-correlations, they require a specialized flagging strategy.

auto-correlations; (v) Only the XX , XY and YY correlations are used for detection, to reduce unnecessary computations. YX correlations are equal to the conjugated XY correlations, and using these for flagging does not provide additional information.

A result of this auto-correlations strategy is shown in the bottom image of Fig. 5. Visual inspection shows that all visible RFI is indeed detected, and the number of false detections appears low. Because we do not have a ground truth, we do not try to quantify these results. Similar to the cross-correlation strategy, the auto-correlation strategy flags parts of the sub-band edges. The centre image of Fig. 5 shows the high-pass filtered data of the final iteration.

2.8. Avoiding HI removal

In observations that cover bright nearby galaxies or the Galactic plane, the 1420 MHz HI line may be detectable in the visibilities from a single cross-correlated baseline. For example, the top-left image of Fig. 6 shows one baseline from a M31 observation, which clearly shows a contribution from HI-emission around 1420 MHz. This poses a challenge for RFI detection, because such a fine, spectrally consistent signal is quite similar to RFI. As shown in the top-right image of Fig. 6, when standard flagging is performed on these data, the HI emission is detected as RFI.

We analyse different ways to mitigate this. In the Netherlands, frequencies between 1400–1427 MHz are reserved for radio astronomy and other forms of passive research¹, and transmitting inside this band is not allowed. As a result, these frequencies are almost free of man-made emission. A simple mitigation strategy is therefore to disable RFI detection inside this band. Unfortunately, the recorded visibilities do occasionally contain strong, non-astronomical values inside this band. The three vertical lines in the images of Fig. 6 are an example of such an observation. Most frequently, these are caused by saturation of a receiver, causing a broadband-like signal in the recorded visibilities, although they might occasionally be caused by RFI emitted at these frequencies (e.g. from a sparking device or lightning). Leaving these broadband contaminants in the data causes degradation of the images. In particular, they cause visible stripes in continuum, full bandwidth images.

Another approach is to flag only based on Stokes Q , U and V . Man-made RFI is often polarized, whereas the sky emission in these polarizations is generally much fainter. The result of this approach is shown in the bottom-left image of Fig. 6. While a part of the HI emission has been left intact, it is still bright enough in these polarizations to get detected. This is even the case when flagging on only one of these polarizations: the HI emission is present in all of the polarizations. Moreover, we occasionally observe RFI that is only visible in Stokes I , and removing any of the polarizations decreases the effectiveness of RFI detection. In Fig. 6, the transmitter around 1425 MHz / 0:00 UTC is for example not as well detected in this approach compared to standard flagging.

Because none of these approaches give good results, we consider another approach, and run the flagger twice: in run A) we flag the data with the normal detection strategy, and in run B) we run the detection with a strategy that is insensitive to spectral lines. For frequencies outside the HI range we use the flags from run A), and inside the HI range (1418–1424 MHz) we use B). The result of this approach is shown in the bottom-right image of Fig. 6. With this approach, broadband structures have been detected as RFI and HI emission is left in the data.

To avoid flagging spectral lines in run B), we adjust the following flagging settings during this run):

- The high-pass filter in frequency direction is set to have a kernel size of one channel, to filter out fluctuations in frequency.
- The sensitivity of the time-direction sumthreshold step is decreased by a factor of 4, to reduce flagging of line-like structures.
- The sensitivity of the frequency-direction sumthreshold step is decreased by a factor of 2. This reduces flagging of temporal fringes in HI emission.

¹ The Dutch spectrum allocations can be found at <https://www.agentschaptelecom.nl/>

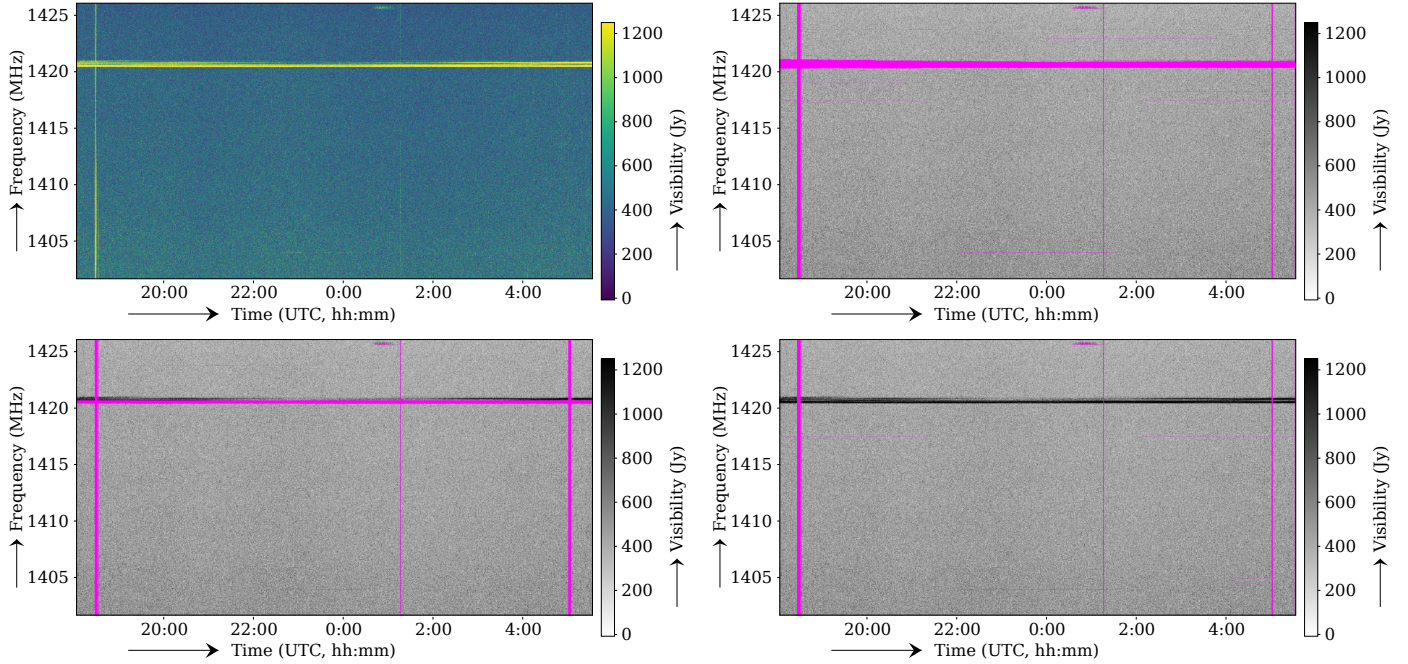


Fig. 6. Band-pass corrected M31 data from WSRT RT9 \times RTA with a strong HI signal. Top-left image: input data. The bright emission around 1420 MHz is from HI and should not be flagged. The vertical lines are instrument or RFI artefacts that should be flagged. Top-right image: after RFI detection without HI modifications, showing in pink what is flagged. Bottom-left image: after RFI detection using Stokes Q , U and V . Bottom-right image: after RFI detection using a specialized strategy for 1418–1424 MHz.

- The number of iterations is increased to remain robust in the presence of strong HI emission.

On overall, the resulting strategy is almost entirely insensitive to spectral-line-like structures. The sensitivity to broadband structures is also reduced because of these changes, but given that this strategy remains sensitive to faint broadband structures such as shown in Fig. 6, we consider this tolerable.

Because run B) requires only a small part of the full bandwidth, the second flagging run is relatively fast, hence the increase in computations caused by this is modest (about 20%).

2.9. Reading overhead and memory considerations

During the AOFlagger stage of the APERCAL pipeline, observations are stored in the Casacore Measurement Set format. In this format, the data of an observation is lexicographically sorted in time, and then in baseline and frequency. While this ordering is suitable for calibration, flagging requires the data baseline by baseline. Unfortunately, the data for a single baseline is spread throughout the file. Therefore, reading a baseline requires reading the file from beginning to end. Because of the block size and caching of storage media, it is inefficient to read the baselines one by one with this approach.

AOFlagger supports three methods for accessing the data:

- Direct reading. In this mode, the data is directly read from the measurement set just before they are needed. Because multiple baselines are processed in parallel using multi-threading, a few baselines are read from the measurement set at once. This mode results in scanning through the input data multiple times, which is computationally costly.
- Reorder before processing. In this mode, the whole measurement set is reordered by baseline, frequency and then time and rewritten to disk in a binary, internal format before processing is started. This results in reading the data only

twice and is generally faster than the direct reading mode, but requires disk space to store the copy of the data.

- In-memory data. In this mode, the whole measurement set is read into memory before starting processing. This results in reading the data only once and is generally the fastest mode, but requires a considerable amount of memory.

Apertif data sets are large and expensive to read: reading the data more than once is undesirable. As a result, the only acceptable reading mode is the in-memory mode. In the particular computing mode where Apercal runs, the amount of memory required by this mode is a considerable constraint, and requires a dedicated node for each flagging operation performed.

Other observatories have solved this issue by integrating AOFLAGGER into a multi-step preprocessing pipeline that stream through the data, split the data in time for flagging and hand these data over part by part to AOFlagger via its application programming interface. Examples of such pipelines are COTTER (Offringa et al. 2015) and DP3 (Van Diepen et al. 2018), which are preprocessing pipelines for the Murchison Widefield Array and the Low-Frequency Array, respectively. In this approach, several tasks (e.g. conversion, phase rotation, flagging, averaging, compression) can be applied with a single read through the data, thereby reducing the read overhead. In the case of Apertif, such a streaming pipeline does not exist. Instead, aoflagger runs as a stand-alone tool inside Apercal.

To solve the memory and reading issue for Apertif, we implemented a time-chunking approach into aoflagger. In this mode, aoflagger reads small chunks in time and flags these independently. This makes it possible to use the memory reading mode, because the data for individual chunks is small enough to fit in memory. It does imply that the algorithm has less information available to do its RFI detection. Therefore, it is important to let time chunks still have a significant size, because AOFlagger would otherwise not be able to find faint RFI, that is persistent

in time, but not detectable in a small chunk. For Apertif, we use a chunk size corresponding to about half an hour of data.

2.10. Use of Lua

Before AOFlagger version 3, AOFlagger strategies were written in the extensible markup language (XML). An XML file specifies a sequence of steps and is interpreted by AOFlagger, and this sequence is executed separately for the data from every baseline. The sequences run multi-threaded, and reading and writing of data is done outside of the strategy. Examples of XML steps are to calculate visibility amplitudes; running SUMTHRESHOLD or SIR operations on the data; or to combine the flags of all polarizations.

Over the years, the use of AOFlagger extended to more and more use-cases: different telescopes, flagging after calibration, high-resolution flagging, etc. It became desirable to make the strategies more flexible. In particular, it became desirable to support standard scripting structures such as loops, conditionals, variables and to provide standardized documentation of the steps. The idea was therefore formed to embed a standard interpreter into AOFlagger and provide a function interface for each step. The data-intensive computations are still performed by high-performance precompiled C++ code, while these are glued together using an interpreted script, thereby combining flexibility with high performance.

Our first approach was to embed it into Python, because of its popularity in astronomical data science. After having implemented a prototype that embeds the Python interpreter into AOFlagger, it turns out some of the features of the Python interpreter conflict with how AOFlagger runs these scripts. Particular challenges were to deal with the global interpret lock; memory management; and fast restarts of the interpreter. While there are various ways to work around these issues, the design goals of the Python language and interpreters do not focus specifically to make the language embeddable.

Lua² is a scripting language that is widely used for embedding scripts in applications, notably in computer games to implement scripted game sequences. This scenario is close to the AOFlagger use-case: the interpreter is integrated into such games, called many times and supports multi-threaded script execution. Algorithmic code that requires high performance can be implemented in compiled languages (C++ in the AOFlagger case). With this idea in mind, we decided to integrate the Lua interpreter into AOFlagger and implement all steps as Lua functions.

The use of a full scripting language has increased the possibilities inside the flagging strategies considerably. For example, it is now possible to adapt the strategy based on properties such as the baseline length, frequency, auto- or cross-correlation, etc. A consequence of the new interface is that existing strategies need to be rewritten, which can not be done automatically. All default strategies have been rewritten to use Lua, which currently includes specialized scripts for 11 observatories (Aartfaac, APERTIF, Arecibo, ATCA, Bighorns, JVLA, MWA, WSRT, LOFAR, NenuFAR). These have all been verified to produce the same result as the old XML-based strategies. Because the new function interface gives better control over what steps need to be run, the speed of the new strategies is slightly higher (several percent). We do not notice any significant overhead from using Lua: the computational time is dominated by the computations inside the function calls.

² <https://www.lua.org/>

3. Results

Apertif observations are processed by the automated Apercal pipeline. This pipeline includes the flagging strategy as described in Sect. 2. In this section, we present results of the full flagging step on Apertif observations. The data that we look at has been recorded between 2019 and 2022. Science products from the first year of observing have been described in the first Apertif data release (Adams et al. 2022; Kutkin et al. 2022).

3.1. RFI detection examples

The detection strategy described in Sect. 2 runs fully automated, and does not require further flagging before calibration and continuum imaging. In general, manual inspection of data after RFI detection shows no residual RFI and few false positives. Figure 7 shows the 1280–1430 MHz range of a typical observation. The top plot shows the data before RFI detection, and the bottom plot shows in white what has been detected as RFI. Figure 8 shows a challenging case with wider bandwidth, with a moderate amount of RFI, missing data (1200–1220 MHz) and strong fringes. Top and bottom plots show again before and after detection. This also demonstrates the challenging situation for radio astronomical science between 1150 and 1300 MHz.

For continuum imaging, it is often useful (or at least pragmatic) to take out any visibility that appears to have a contribution from RFI. For spectral imaging, a flagging result such as shown in Fig. 8 is problematic, because many channels are fully removed. In those cases, it is possible to reduce the sensitivity of the RFI detection. The sensitivity is specified as a variable in the script. For the detection result shown in Fig. 9, the sensitivity was decreased by a factor of 3. Compared with the result in Fig. 8, this reduced the flagging from 49 to 33%. This takes out the strongest RFI, but leaves weak (but visible) RFI in the data. Decreasing the sensitivity further continues to trade the availability of visibilities with a lower quality of those visibility.

3.2. RFI characteristics and long-term statistics

During the flagging step, statistics are collected that summarize the (detected) RFI occupancy and data quality. We have collected these statistics for 304 of the currently processed observations. Averaged over all these observations and the full bandwidth, the total detected RFI occupancy is 11.1% in the cross-correlated baselines and 14.6% in auto-correlated baselines. Figure 10 shows the detected spectral RFI occupancy for each observation, as well as the occupancy averaged over all observations. Only cross-correlated data is included. At most frequencies, the average loss of data due to RFI is about 10%, but with a spread of approximately 0–15% between observations, and a few larger outliers.

Frequencies between 1400 and 1427 MHz are reserved for radio astronomy. At these frequencies, the average RFI occupancy is slightly lower (approximately 8%), but is evidently still affected by instrumental effects (such as receiver saturation) or natural and unintended RFI (such as lightning). Figure 6 shows data that is affected by such broadband artefacts. It is likely that the ~10% base-level of occupancy is caused by such artefacts.

Some observations show a small excess RFI occupancy at 1420 MHz. This is caused by HI that is detected as RFI. The methods to avoid flagging HI that are described in Sect. 2.8 were implemented only halfway 2021. Some of the observations that are flagged before that still show false-positive detections

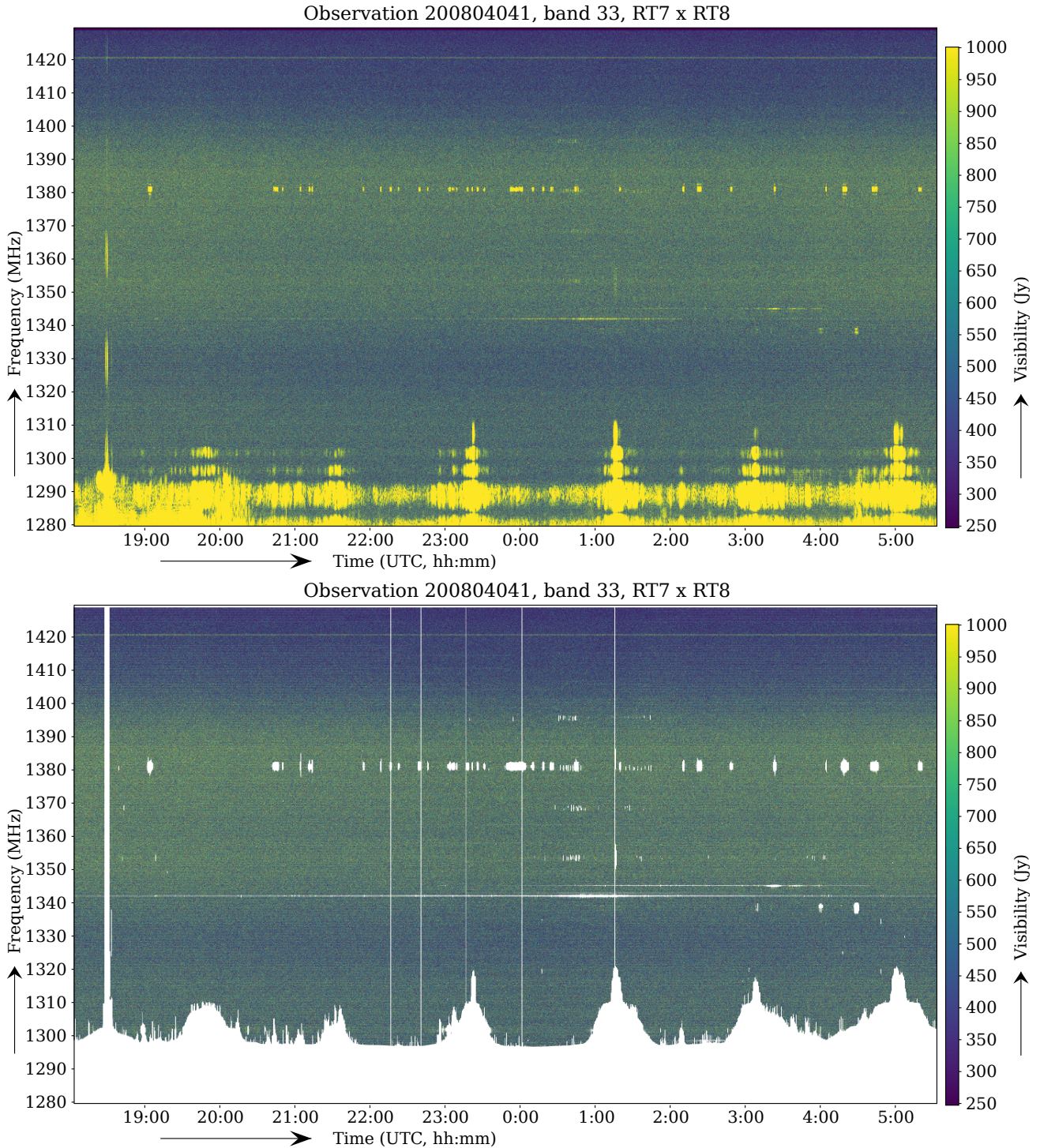


Fig. 7. Typical flagging result for a single baseline in a wideband observation. The top panel shows the input visibilities, and the bottom panel shows the visibilities overlaid with the detection result in white. These plots show the Stokes I visibilities. Some interference features are only visible in Stokes Q , U or V , such as the vertical features around midnight. All interference features have successfully been detected, and no obvious undesirable detections are visible, with the exception of horizontal flagged features every 200 kHz, caused by the sub-band bandpass (see Fig. 4). 18% of the data gets flagged for the baseline in this observation.

at HI frequencies, but all observations after avoiding HI was implemented show indeed no HI flagging.

The same base level of 10% is not visible at frequencies above 1430 MHz. The reason for this difference is that only a relative small number of observations cover frequencies above 1430 MHz. Frequencies between 1427 and 1492 MHz are allocated to various services, including mobile communication and

fixed transmissions³. Some of these are satellite based. In 2020, the 1452–1492 MHz band was auctioned in the Netherlands and thereafter allocated for the use of 5G mobile phone downlink. As shown in Fig. 10, the use of data above 1430 MHz is limited.

³ See <https://www.agentschaptetelecom.nl/>

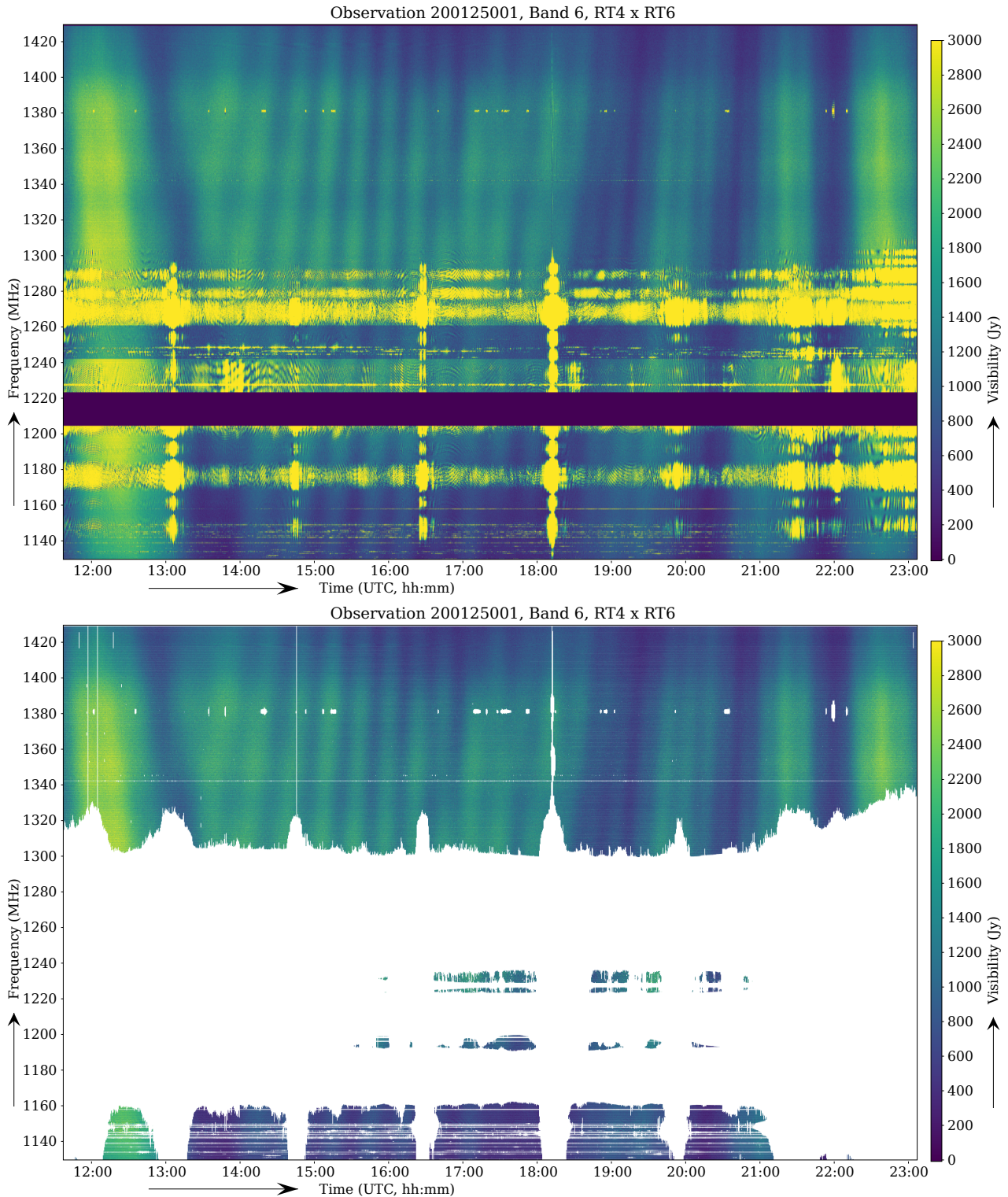


Fig. 8. Detection result for a full 300-MHz bandwidth observation.

Some channels between 1300–1400 MHz contain a few outlier RFI occupancies. These are caused by a nearby radar station that is occasionally turned on. Frequencies between 1130 and 1300 MHz are predominantly affected by RFI from Global Navigation Satellite Systems (GNSS), such as the US GPS, Russian GLONASS, Chinese BeiDou, and European Galileo satellite constellations. All these constellations use satellites in orbits at ~ 2000 km and with high orbital inclinations ($i = 54\text{--}65^\circ$)

to provide global coverage. Frequencies for wide band transmissions are assigned to, and shared between, these systems at 1176.45, 1191.795, 1207.14, 1227.6, 1278.75 MHz (for GPS, BeiDou, Galileo) and 1202.025 and 1242.9375–1251.6875 MHz (for GLONASS).

Wide band signals are detected at these frequencies throughout the entire observation of Fig. 8 covering the band down to 1130 MHz. Using orbital ephemerides of these satellite

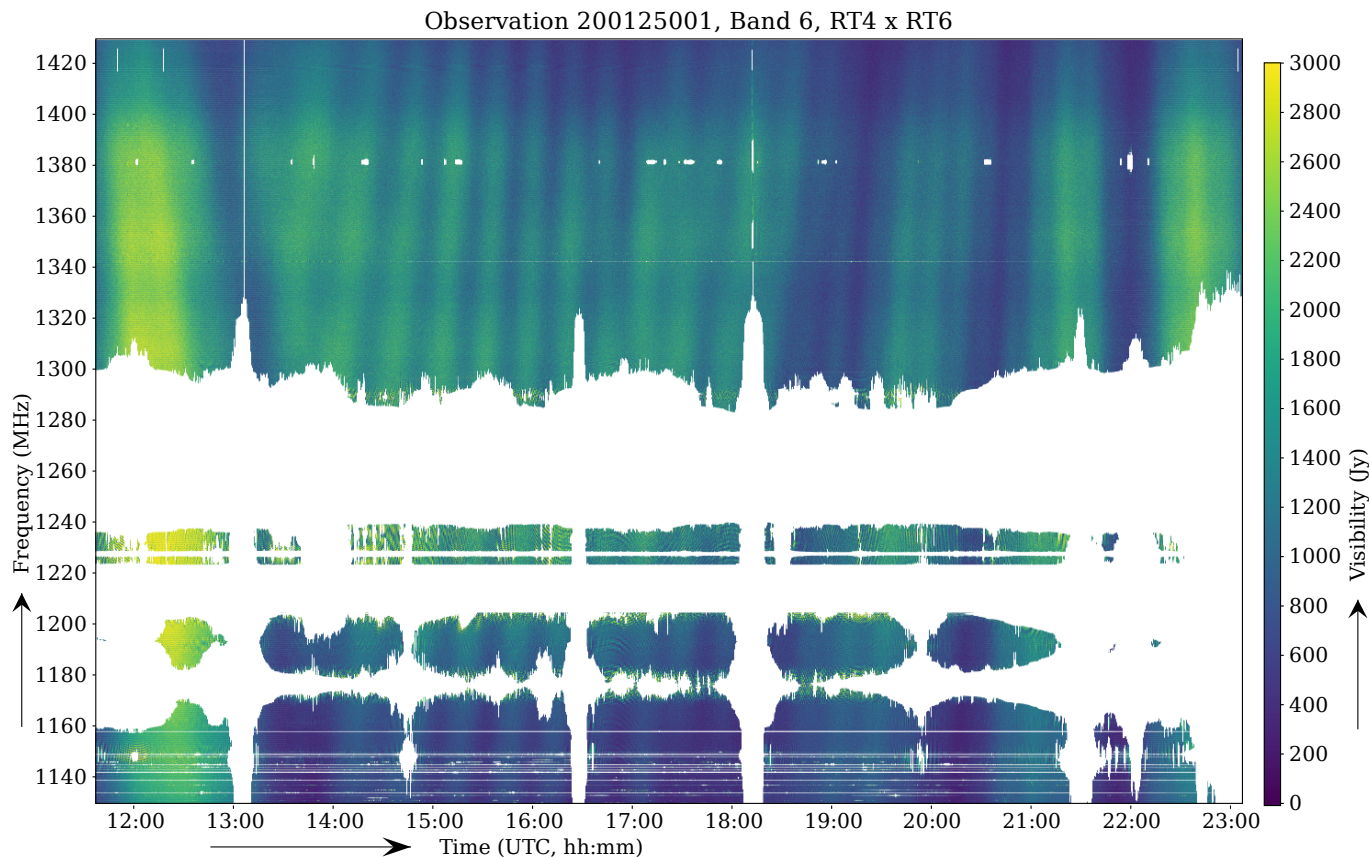


Fig. 9. Same as Fig. 8, but flagged with 3× lower sensitivity.

constellations, we find that the strong temporal RFI observed in Fig. 8 at 13:06, 14:46, 16:29, 18:13 and 19:54UTC is caused by BeiDou satellites passing within 5° from the pointing of the APERTIF compound beam. The pass of 18:13UTC had a minimum separation of $0^\circ.31$ and led to saturation of the receiver, affecting the entire observing band. Two GPS satellites passed at $1^\circ.47$ and $2^\circ.30$ separation from the beam pointing at 22:02 and 23:02UTC, and one Galileo satellite at $3^\circ.72$ at 22:59UTC, and coincident increases of the RFI levels are observed, but not as strong as with the passes of BeiDou satellites. The GNSS signals observed away from these passes near the primary APERTIF beam are likely due to far sidelobes or multi-path reflections of GNSS signals from the WSRT focus structure or other nearby structures directly into the receiver.

3.3. Computational requirements

In this section we summarize the computational requirements of the Apertif RFI detection strategy, with the aim of making it possible to approximate the computational requirements for other telescopes when a similar flagging strategy is used. Since the total throughput is depending on many complex factors of the computing platform (e.g. clock speed, cores, memory bandwidth, instruction set, vectorization), we aim at giving a first-order estimate only.

We measure the performance of flagging a set with visibilities from a single observation. We use an Apertif observation with 1346 timesteps, 24 572 channels and 4 polarizations, for a total of 132M visibilities. This makes the visibility data, which consists of 4-byte single-precision real and imaginary values, 1.1 GB in size.

We perform our test on a desktop machine with an AMD Ryzen 7 2700X 8-Core processor and 64 GB of memory. This processor can perform hyper-threading, and thus we run 16 detections in parallel. We load the data in memory before detection and do not store the results, to avoid any disk access. Averaged over 10 runs, it takes 46 s to run 16 detections, which amounts to a throughput of 370 MB/s (or 46M visibilities/s). At the time of writing, a typical fast spinning disk achieves a sustained reading throughput of a few hundred MB/s. Hence, disk access can be a significant cost of a stand-alone RFI detection step. This can be problematic for supercomputers, because they have high computing power, but not a high I/O throughput.

3.4. Comparison against a machine learning approach

Some studies have found that machine learning can improve the accuracy of RFI detection. In Yang et al. (2020), the authors test their own SUMTHRESHOLD implementation against a machine learning approach, using a ground truth flag mask that is manually determined by an engineer. Such a ground truth mask is difficult to make in general, including for Apertif data, where broadband RFI tapers off and it is unclear from which points samples are truly unaffected by RFI. We can however conclude that, after our pipeline, all visibly affected samples have been identified. Moreover, imaging results have achieved the thermal noise of the instrument, thereby indicating that the accuracy of interference detection is not a limitation.

This conflicts somewhat with the conclusions made by Yang et al. (2020). The SUMTHRESHOLD implementation that is used there to compare their results with, does not achieve the

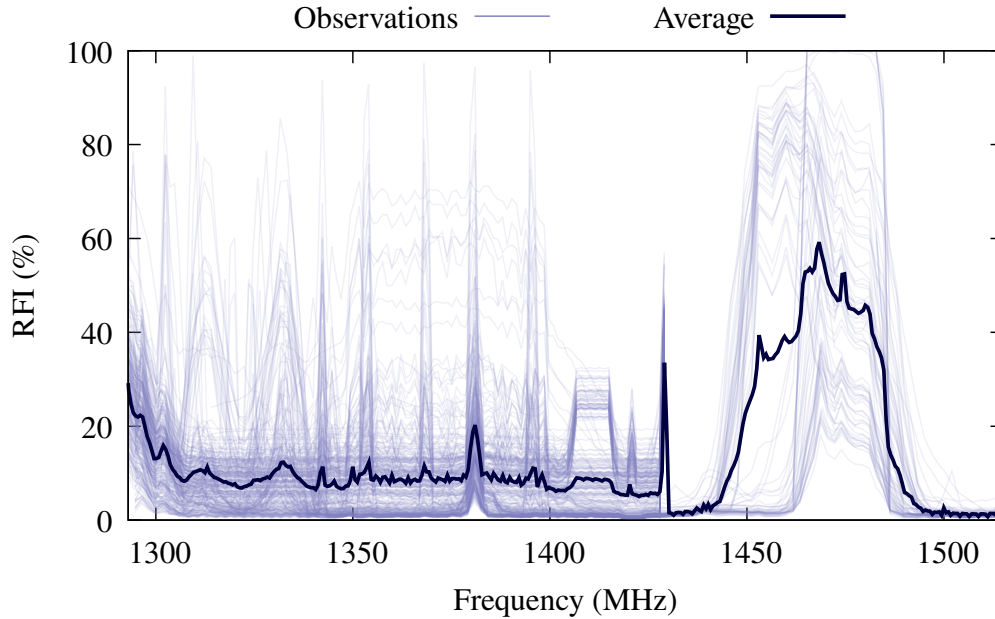


Fig. 10. Percentage of RFI over frequency detected in 304 Apertif observations, excluding auto-correlations.

published accuracy of AOFLAGGER, because residual interference is visually present. Potential explanations for these differences could be (i) that Yang et al. train their network for a specific scenario but did not optimize their SUMTHRESHOLD approach; or (ii) that they do not use a full (i.e. AOFLAGGER-like) SUMTHRESHOLD-based pipeline that includes the SIR operation and that is similarly optimized for their instrument. An important consideration is that morphological operations are aimed at detecting RFI that is below the noise, therefore invisible to scientists that manually classify RFI. In the comparisons done in Yang et al. 2020, samples detected by the morphological operator would all be counted as false positives, whereas this operator has been shown to improve the final science results (Offringa et al. 2012). It can therefore not yet be stated that, based on accuracy, machine learning methods are outperforming traditional based methods. Rather, it is clear that both methods are competitive and are accurate enough to largely mitigate the problem of interference in radio data.

There are differences in the computational performance though. In Xiao et al. (2022), machine learning methods flag a one-hour FAST observation of 67 GB in 61% of the observing time using 8 computing nodes (Xiao et al. 2022). This amounts to a single-node computational performance of 14 GB/h. On the other hand, the single-node performance of the AOFLAGGER approach listed in Sect. 3.3 is 370 MB/s, or 1.3 TB/h, and AOFLAGGER is therefore almost two orders of magnitude faster. While the performance of the computing nodes used for the computational performance analyses may differ somewhat, and it is therefore not a direct comparison, it is evident that the AOFLAGGER approach is significantly faster. In Sun et al. (2022), authors compare the run-time of AOFLAGGER to their convolutional neural network (CNN) approach and find that AOFLAGGER is two to four times faster. However, the authors measured the total run-time of the aoflagger executable, which would include disk access, start-up overhead and time spent in the CASACORE library to transfer the measurement set data. Because the flagging speed is near the disk access speed, this overhead can be substantial. A better benchmark is possible by using the C++ or Python API of AOFLAGGER directly. On their Sim_RFI-1

dataset, they reach an AOFLAGGER speed of 250 GB/h, while in this work, with a more advanced strategy, we reach 1.3 TB/h on similar hardware. Their CNN method reaches a speed of 145 GB/h, which is an order of magnitude faster than what is reached by Xiao et al. (2022), but is an order of magnitude below what we reach with our AOFLAGGER approach.

4. Discussion and conclusions

We have described and demonstrated an automated RFI detection strategy aimed at flagging Apertif data. Our detection strategy implements novel SUMTHRESHOLD and SIR-operator algorithms that take prior information about invalid data into account. It also avoids the flagging of HI emission, works on auto-correlations, corrects the sub-band band-pass and contains some further parameter optimizations for Apertif. The change from the AOFlagger XML strategies towards fully scripted strategies provides flexibility that made these changes quite easy to implement and supports flexibility during experimentation. Besides making the process easier and faster, an automated RFI detection strategy also makes the results reproducible, compared to when RFI is flagged manually, and it allows reducing the data size by averaging early on in the data reduction processing.

We expect that our RFI detection strategy will work for data from other instruments, in particular those with a frequency coverage comparable to Apertif, such as MeerKAT, ASKAP, JVLA and future SKA-mid observations around 1.0–1.5 GHz. Different bands might require some changes to the strategy parameters, but should be able to reuse a large part of the approach.

While machine learning techniques may compete with the accuracy of AOFlagger, they do not compete with its speed. Moreover, we have shown it is possible to add new features to AOFlagger, such as avoiding the 21-cm HI signal, accurate detection in the presence of invalid data and flagging of auto-correlations. None of the current available machine learning techniques support these scenarios. Most parameters, such as the sensitivity towards broadband and line RFI, or the expected

smoothness of the data, are intuitive and easy to tweak for science cases that, for example, require that transients do not get flagged, or that require a difference balance between taking out all visible RFI on one hand, and keeping as much data available for further processing on the other hand. This will be challenging, if at all possible, to implement in a machine learning framework.

In this work, we have not made use of the multi-beaming capabilities of Apertif: beam are flagged independently. While some first-order testing indicates that using data integrated over all beams does not improve flagging accuracy, it can be expected that RFI does correlate somewhat over beams. A strategy where the integrated data is searched for RFI, and where this is used as additional input for the flagging of individual beams, might be effective for detecting RFI that is below the noise for a single beam.

Acknowledgements. This work makes use of data from the Apertif system installed at the Westerbork Synthesis Radio Telescope owned by ASTRON. ASTRON, the Netherlands Institute for Radio Astronomy, is an institute of the Dutch Research Council (de Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO). B.A. acknowledges funding from the German Science Foundation DFG, within the Collaborative Research Center SFB1491 “Cosmic Interacting Matters – From Source to Signal”. E.A.K.A. is supported by the WISE research programme, which is financed by NWO. J.M.vd.H. and K.M.H., acknowledge funding from the European Research Council under the European Union’s Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement No. 291531 (‘HISStoryNU’). Jv.L., Y.M. and L.C.O. acknowledge funding from the European Research Council under the European Union’s Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement No. 617199 (‘ALERT’; PI: JvL). K.M.H. further acknowledges financial support from the State Agency for Research of the Spanish Ministry of Science, Innovation and Universities through the “Center of Excellence Severo Ochoa” awarded to the Instituto de Astrofísica de Andalucía (SEV-2017-0709) from the coordination of the participation in SKA-SPAIN, funded by the Ministry of Science and innovation (MICIN) and grant RTI2018-096228-B-C31 (MCIU/AEI/FEDER,UE). Jv.L. further acknowledges funding from Vici research programme ‘ARGO’ with project number 639.043.815, financed by NWO. D.V. acknowledges support from the Netherlands eScience Center (NLeSC) under grant ASDI.15.406.

References

- Adams, E. A. K., Adebahr, B., de Blok, W. J. G., et al. 2022, *A&A*, **667**, A38
 Adebahr, B., Schulz, R., Dijkema, T., et al. 2022, *Astron. Comput.*, **38**, 100514
 Fridman, P. A. 2008, *AJ*, **35**, 1810
 Gary, D. E., Liu, Z., & Nita, G. M. 2010, *Proc. Science*, RFI2010
 Harrison, K., & Mishra, A. K. 2019, in *2019 RFI Workshop - Coexisting with Radio Frequency Interference (RFI)*, 1
 Hellbourg, G., Weber, R., Abed-Meraim, K., & Boonstra, A. 2014, in *Acoustics, Speech and Signal Processing (ICASSP)*, 2014 IEEE International Conference on, 5387–5391
 Kocz, J., Briggs, F. H., & Reynolds, J. 2010, *AJ*, **140**, 2086
 Kocz, J., Bailes, M., Barnes, D., Burke-Spolaor, S., & Levin, L. 2012, *MNRAS*, **420**, 271
 Kutkin, A. M., Oosterloo, T. A., Morganti, R., et al. 2022, *A&A*, **667**, A39
 Middelberg, E. 2006, *PASA*, **23**, 64
 Offringa, A. R., de Bruyn, A. G., Biehl, M., et al. 2010a, *MNRAS*, **405**, 155
 Offringa, A. R., de Bruyn, A. G., Biehl, M., & Zaroubi, S. 2010b, *Proc. Science*, RFI2010
 Offringa, A. R., van de Gronde, J. J., & Roerdink, J. B. T. M. 2012, *A&A*, **539**, A95
 Offringa, A. R., de, A. G., Zaroubi, S., et al. 2013, *A&A*, **549**, A11
 Offringa, A. R., Wayth, R. B., Hurley-Walker, N., et al. 2015, *PASA*, **32**, e008
 Peck, L. W., & Fenech, D. M. 2013, *Astron. Comput.*, **2**, 54
 Prasad, J., & Chengalur, J. 2012, *Exp. Astron.*, **33**, 157
 Purver, M., Bassa, C. G., Cognard, I., et al. 2021, *MNRAS*, **510**, 1597
 Sclocco, A., Vohl, D., & van Nieuwpoort, R. V. 2019, in *2019 RFI Workshop - Coexisting with Radio Frequency Interference (RFI)*, 1
 Sun, H., Deng, H., Wang, F., et al. 2022, *MNRAS*, **512**, 2025
 Taylor, J., Denman, N., Bandura, K., et al. 2019, *J. Astron. Instrum.*, **08**, 1940004
 Tingay, S. J., Goeke, R., Bowman, J. D., et al. 2013, *PASA*, **30**, 21
 van Cappellen, W. A., Oosterloo, T. A., Verheijen, M. A. W., et al. 2022, *A&A*, **658**, A146
 van Diepen, G., Dijkema, T. J., & Offringa, A. 2018, *Astrophysics Source Code Library* [[record ascl:1804.003](https://doi.org/10.26434/chemrxiv-2018-0003)]
 van de Gronde, J. J., Offringa, A. R., & Roerdink, J. B. T. M. 2016, *J. Math. Imaging Vision*, **56**, 455
 van Haarlem, M. P., Wise, M. W., Gunst, A. W., et al. 2013, *A&A*, **556**, A2
 Wilensky, M. J., Morales, M. F., Hazelton, B. J., et al. 2019, *PASP*, **131**, 114507
 Winkel, B., Kerp, J., & Stanko, S. 2006, *Astron. Nachr.*, **88**, 789
 Xiao, J., Zhang, Y., Zhang, B., et al. 2022, *New Astron.*, **96**, 101825
 Yang, Z., Yu, C., Xiao, J., & Zhang, B. 2020, *MNRAS*, **492**, 1421