Handbook WSRT flagger

version 0.61

by Adriaan Renting

document version 0.4

24-07-2006

Table of Contents

1 Introduction

The WSRT flagger is a small tool written in C++ to flag visibility data points containing interference in AIPS++ measurement sets. It has two mayor differences with most currently available flagging tools.

- First it uses the complex values of the visibilities directly, instead of amplitude or phase only.
- Secondly it has a mechanic to flag time slots based on the root mean square (RMS) value of the remaining unflagged data points.

Within the context of this discussion flagged means a visibility data point is marked as invalid for inclusion in further computations. Interference can be radio interference from an external terrestrial source, like for example a mobile phone, military aircraft, TV-station, or interference can be from an internal source like for example a malfunctioning or inoperative telescope or correlator.

1.1 Purpose

The main reason that the WSRT flagger was written is that there is currently no other tool capable of detecting correlator errors in WSRT data. to be able to detect these errors first the worst RFI needs to be flagged, leading to the current two layered approach.
The first step is to flag interference using a new algorithm based a complex median on the data. The second step tries to apply a couple of constraints on the RMS values of the remaining unflagged data to detect correlator errors. These limits are both based on theoretical upper and lower bounds and on some empirical knowledge about how the WSRT operates. Most of the algorithms and constraints should also be valid for other interferometers.
The flagger could be useful for three types of WSRT users:

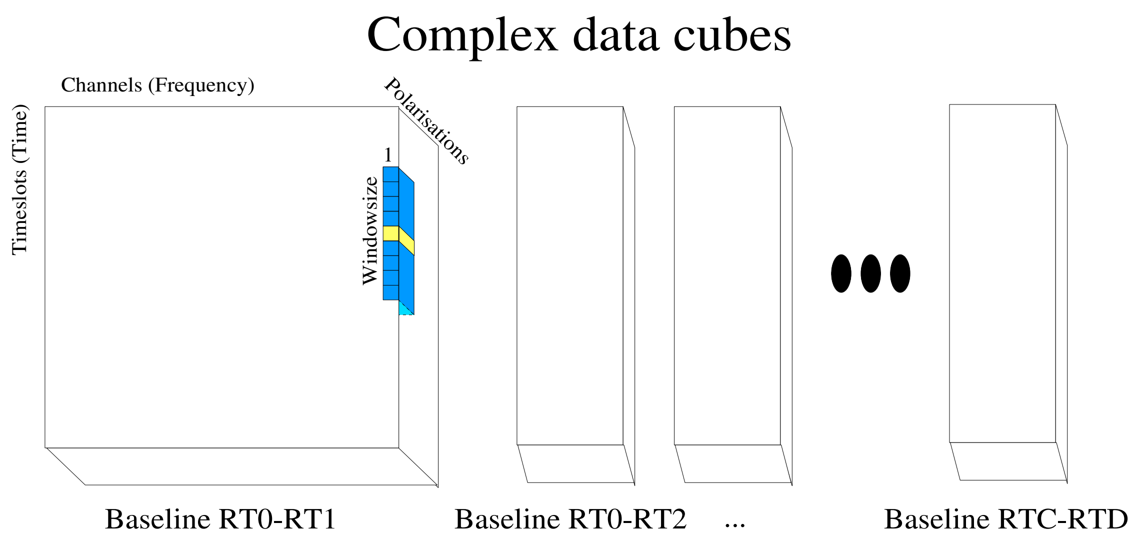
- The operational staff can use it as a rough detector of RFI and correlator errors.
- The inhouse astronomers can use it when having a first look at measured data.
- External astronomers can use it to flag interference and errors in their WSRT datasets.

2 Algorithm

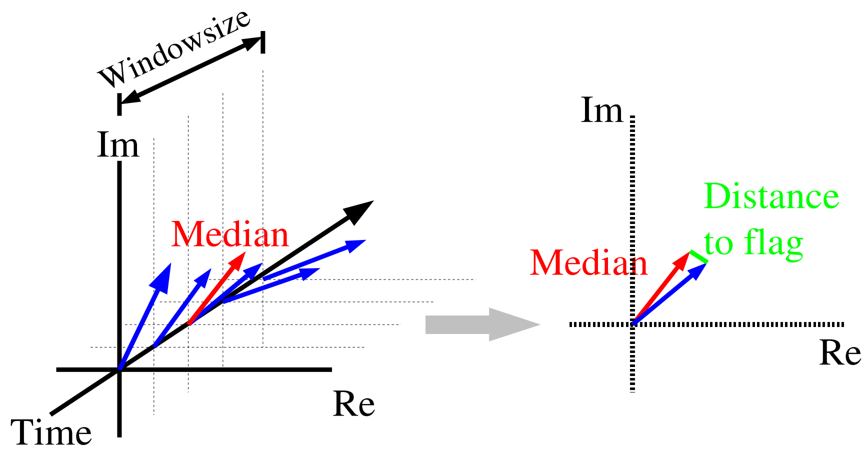
The flagger consists of two algorithms:

- A basic algorithm, which is used to inspect single data points in the measurement to determine their validity. This algorithm will mainly detect interference.
- A second algorithm that uses root-mean square statistics of an entire timeslot to determine if a complete baseline or telescope is malfunctioning. This will usually be due to correlator errors.

The basic algorithm handles each combination of baseline, channel and polarization separately.



For each of these combinations it iterates over the time slots in the measurement set and computes the complex median of a sliding window of size Windowsize. Except for the first and last Windowsize/2 values, the window is centered on the point to be inspected for flagging. The first and last Windowsize/2 values get inspected for flagging based on a window whose edge is either the first or last time slot.



The data point under inspection gets flagged if it's absolute complex distance to the median of the window is above a baseline dependent threshold.

$$flag\ point\ = \left|inspected\ point - complex\ median\right| > Baseline\ dependent\ threshold$$

If one polarization of a data point exceeds this threshold, all polarizations will be flagged.

The baseline dependent threshold is basically a lower estimate for the noise level multiplied by a factor that scales linearly from a user supplied value min at baseline length 0, to the user supplied value max at maximum baseline length.

$$Baseline\ dependent\ threshold\ = Noise_{Estimate} \cdot \left(min + \left(\frac{Baseline\ length}{Maximum\ baseline} \cdot (max - min) \right) \right)$$

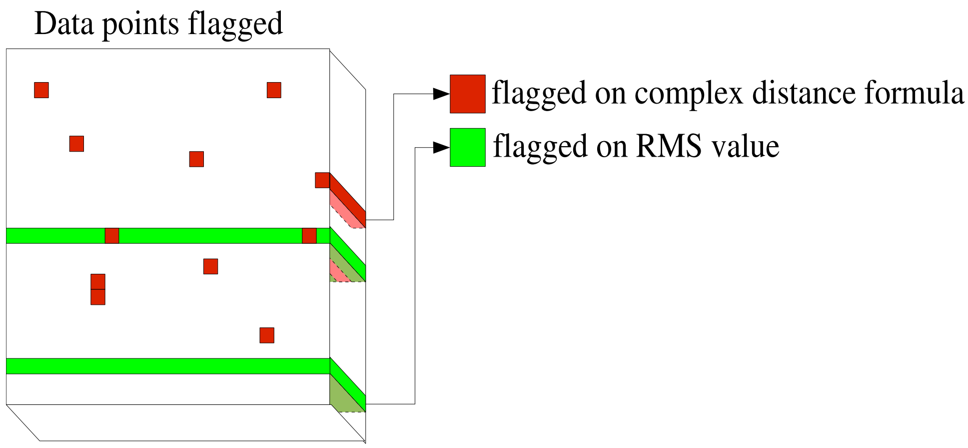
The threshold is baseline dependent because at longer baselines the threshold might need to be higher to avoid flagging interferometer fringes.

The lower estimate for the noise level is the inverse of the square root of the integration time multiplied with the bandwidth of a single channel.

$$Noise_{Estimated} = \frac{1}{\sqrt{t_{integration} \cdot Bandwidth_{channel}}}$$

The min and max parameters of the flagger are therefore expressed in units of this theoretical noise estimate. This formula gives a theoretical lower limit, which in practice will not be reached. The reason the noise estimate is nevertheless computed using this formula, is because it gives a limit that is independent of the observed sources and interference.

Most other flagging algorithms use a threshold based on a mean or median of the actual data. This would result in a biased threshold, if a lot of interference is present, and less accuracy if the observed source is strong.



Next to the complex distance based flagging, there is a second mechanism based on a RMS computation. For each baseline-timeslot combination the RMS is computed of all points that are not flagged with the complex distance algorithm. If the RMS value of such a baseline-timeslot slice is above 1.5 or below 0.5*the noise estimate, the entire time-baseline slice is flagged.

$$Flag\,slice = not\left(\frac{Noise_{Estimated}}{2} < \sqrt{\frac{\sum |value|^2}{N}} < 1.5\right)$$

The computed RMS is also compared to the median RMS of all other telescope-pairs, and flagged if it deviates to much:

$$Flag\,slice = not\left(\frac{median}{10} < \sqrt{\frac{\sum |value|^2}{N}} < median \ x \ 10\right)$$

Finally there is a third mechanism: The autocorrelations get flagged using the complex distance algorithm, with a threshold of 30*$Noise_{Estimated}$, and the RMS algorithm.

$$flag\ point_{autocorrelation} = |inspected\ point - complex\,median| > Threshold_{autocorrelations}$$

$$Threshold_{autocorrelations} = 30 * Noise_{Estimated}$$

If the result of this is that the entire timeslot gets flagged, then for this timeslot all cross correlations containing the autocorrelated telescope are marked as flagged.

3 Usage

3.1 Arguments

The name of the executable is WSRT_flagger. It has the following command options:

- **ms:** The measurement set to be flagged.
- **window:** The window size for the used median.
- **crosspol:** Will only check the cross polarizations to determine if a point is valid.
- **min:** This is the user determined flag threshold at baseline length 0.
- **max:** This is the user determined flag threshold at maximum baseline length.
- **flagrms:** This setting determines if points that would be flagged on RMS or autocorrelation are written to the measurement set.
- **flagdata:** This setting determines if points that would be flagged on complex distance are written to the measurement set.
- **existing:** This setting determines if existing flags on the data will be kept or discarded.
- **data:** This setting determines if the DATA, CORRECTED_DATA, or MODEL_DATA column will be flagged. It is not yet possible to flag residual data.

If the executable is called with incorrect, or without options a help text will be shown.

Example:

WSRT_flagger ms=test.MS crosspol=true window=13 min=4.5 max=6 flagrms=true flagdata=true existing=true data=DATA

Only the **ms** option is mandatory, all other options have default settings, which are shown if the WSRT_flagger is called without the ms argument.

3.2 Compilation

The sources of this program can be found in the CVS tree under users/renting/WSRTflagger. With the AIPS++ build environment properly initialized with aipsinit, you should be able to just run make in this directory to create the executable. You might need to change some variables in the Makefile, like compiler or library locations.

3.3 Caveats

Some points to keep in mind when using the WSRTflagger: The flagger honors changes of field, pointing and frequency. This however means that if the time between such changes in smaller as the window size, the algorithm will fail.
The flagger does not honor existing flags at present.
The flagger will ignore baselines longer as 3000km, as the WSRT often reports a non-existing 15^{th} and 16^{th} telescope with a baseline length of 3854243 meter.
The flagger assumes all exposure times are the same length. It only computes $Noise_{Estimated}$ once.

3.4 Useful settings

In the table below are some settings for the flagger that are known to produce reasonable results.

Data type	***Crosspol***	***window***	***min***	***max***
Faint sources 21cm data, no XY and YX	false	7	5	6
Bright sources 21cm data, no XY and YX	false	13	10	15
Bright sources 92cm data, with XY and YX	false	7	10	15
Bright sources 92cm data, with XY and YX	true	3	4	6
92cm data, with XY and YX	true	7	12	14
LFFE data, with XY, YX	true	25	3	3

FlagRMS mainly checks for correlator errors, but sometimes also flags channels if there is a lot of RFI, usefulness is therefore largely not related to the type of data observed. Especially at the lower frequencies it might do you more harm than good.

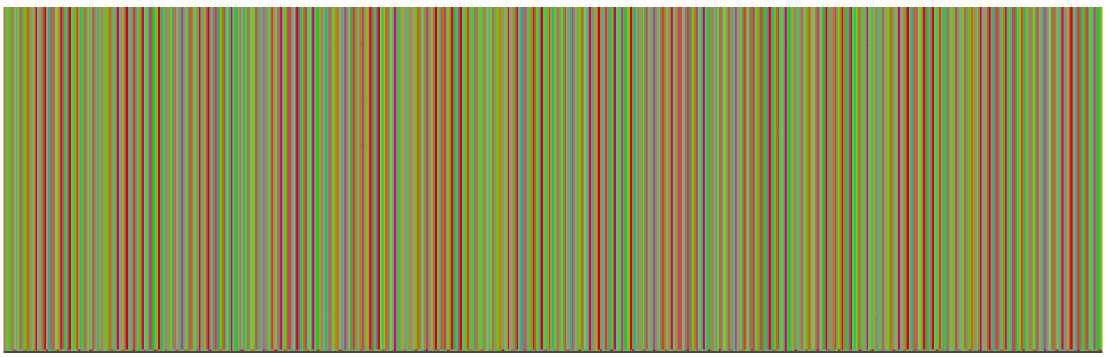
4 Examples

4.1 Example 1

This example applies the WSRT_flagger to a dataset with one malfunctioning crosscorrelation RT6-RT8YY, and a little interference.

Channels (frequency)

Time (30s intervals)

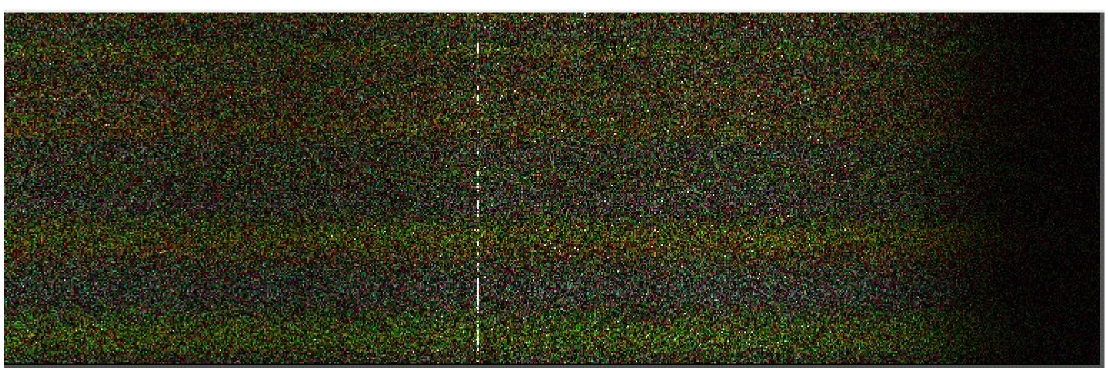


baseline RT6-RT8YY scale:0-32 (arbitrairy units)

The above graph shows the crosscorrelation RT6-RT8YY of measurement 10403049. Intensity denotes amplitude, color indicates phase. When compared to RT6-RT8XX in the image below is is clear that the above data is invalid.

Channels (frequency)

Time (30s intervals)



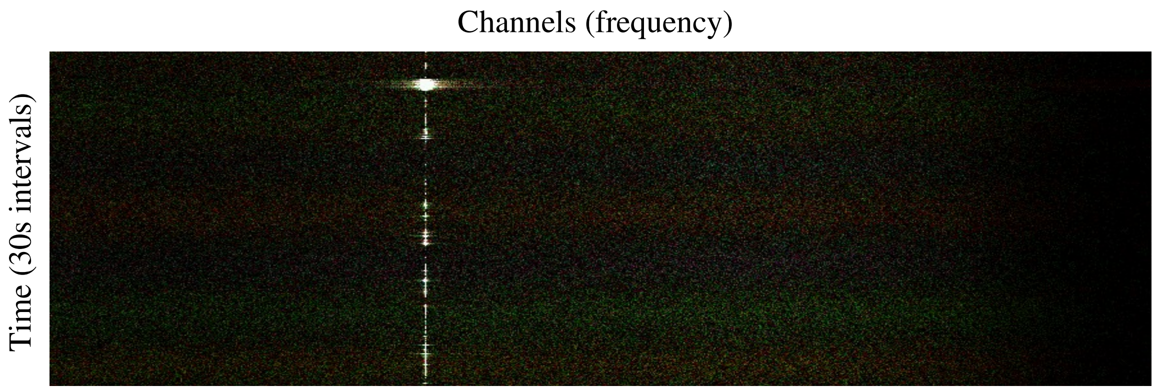
baseline RT6-RT8XX scale:0-0.01 (arbitrairy units)

When using the WSRT_flagger on the above dataset, the problem in RT6-RT8YY is indeed detected and the crosscorrelation RT6-RT8 is completely flagged, while almost none of the other data is flagged.

The output of the WSRT_flagger shows that the correlation RT6-RT8 contains 100% faulty data. It also shows that the rest of the data is almost clear of interference.

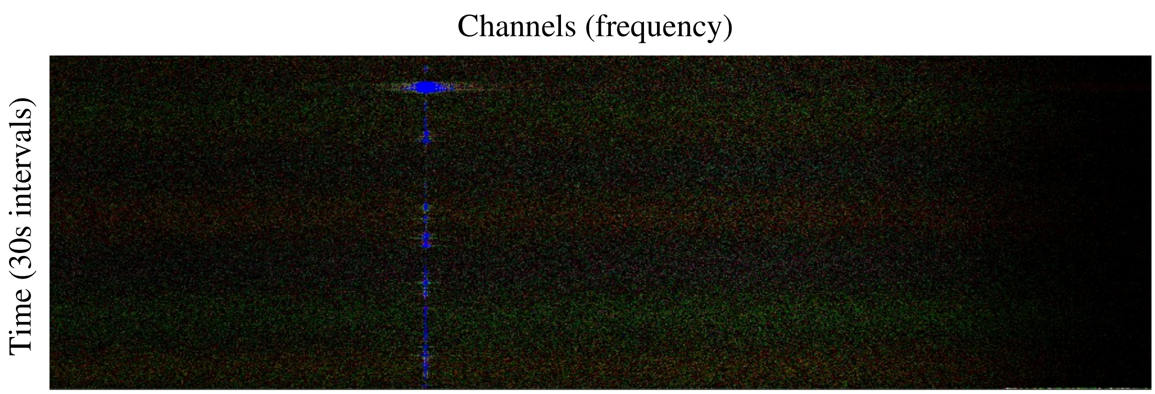
```
14:51-373> WSRT_flagger ms=10403049_S0_T0.MS crosspol=false window=13 min=5 max=5 flagrms=true flagdata=true
Runnning flagger please wait...
WSRT_flagger: Version 0.41 WSRT flagging by Adriaan Renting and Michiel Brentjens 0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Spectral window: 1
       RT0   RT1   RT2   RT3   RT4   RT5   RT6   RT7   RT8   RT9   RTA   RTB   RTC   RTD
RT0    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT1    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT2    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT3    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT4    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT5    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT6    0%    0%    0%    0%    0%    0%    0%    0%    100%  0%    0%    0%    0%    0%
RT7    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RT8    0%    0%    0%    0%    0%    0%    100%  0%    0%    0%    0%    0%    0%    0%
RT9    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RTA    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RTB    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RTC    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
RTD    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%    0%
Spectral windows (flagged %): SW0
                               1%
Antennae (flagged %):  RT0  RT1  RT2  RT3  RT4  RT5  RT6  RT7  RT8  RT9  RTA  RTB  RTC  RTD
                       0%   0%   0%   0%   0%   0%   7%   0%   7%   0%   0%   0%   0%   0%
Elapsed: 2:48.33   CPU: 128.967u+24.410s = 91.1%
```

Next to the problem in RT6-RT8 there is very little interference in 10403049. There is one channel that has some interference, which varies in intensity, and also leaks into adjacent channels in some timeslots.



baseline RT6-RT7XX scale:0-0.05 (arbitrary units)

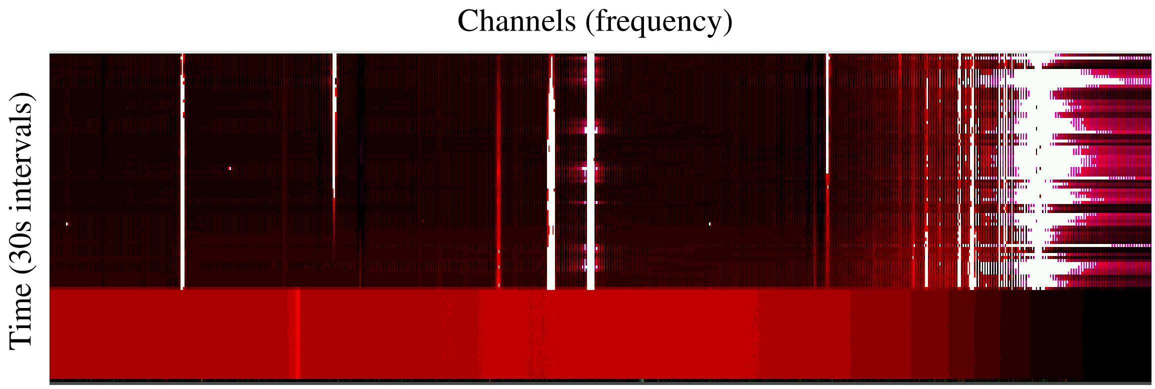
This interference also gets flagged in addition to RT6-RT8, but only the channel-time points that actually contain interference, as shown in the example below, where the color blue indicates flagged data.



baseline RT6-RT7XX scale:0-0.05 (arbitrary units)

4.2 Example 2

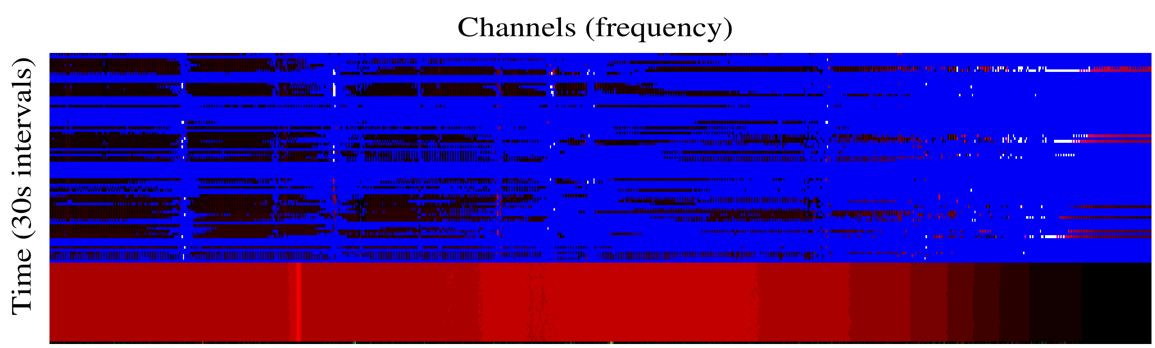
The second example is measurement 10406251, which contains a telescope RTC that malfunctions during part of the measurement. This is most clearly visible in the autocorrelation of RTC:



baseline RTC-RTCXX scale:0-2 (arbitrary units)

In the above image about the last 30% of the measurement does seem to be valid data. When this autocorrelation is flagged, you can see most of the faulty data gets flagged. As described earlier this is with a fixed threshold that is 30 times the theoretical noise level, because this is an autocorrelation. The flagged data is shown in blue im the next image:

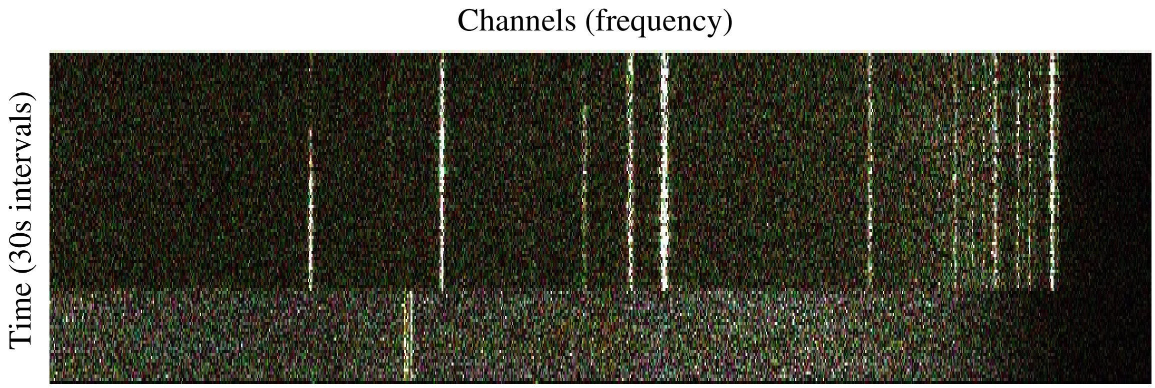
baseline RTC-RTCXX scale:0-2 (arbitrary units)



baseline RTC-RTCXX scale:0-2 (arbitrary units)

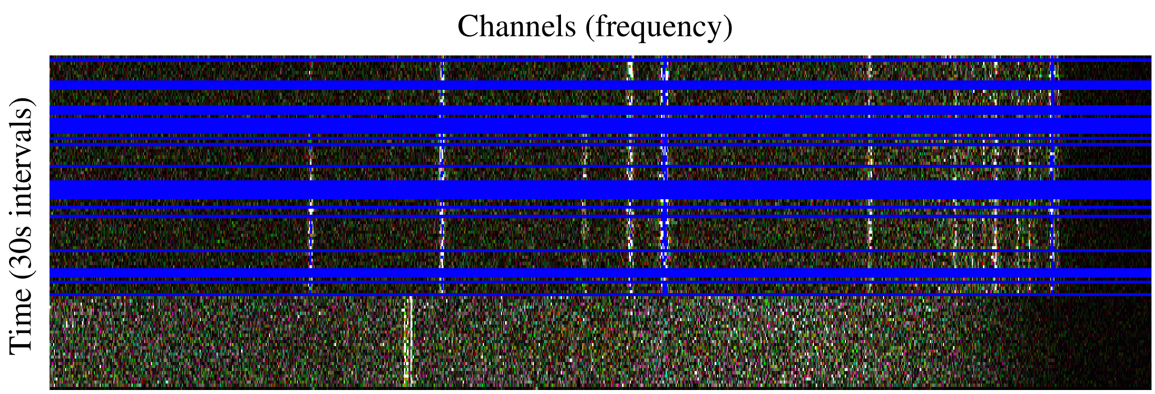
The image shows that none of the celestial signal in the last 30% of the measurement gets flagged while over 80% of the faulty data gets flagged, even with a threshold of 30.

For about half of the faulty timeslots all data points are flagged, which has the consequence that all crosscorrelations containing RTC also get flagged for that specific timeslot. For example the baseline RTC-RTD shown in the image below is clearly suffering from the same problem as RTC-RTC:



baseline RTC-RTD YY scale:0-0.002 (arbitrary units)

The data points being flagged are shown in the image below in blue. The image shows that about 25% of the timeslots get flagged completely because these are also blanked out in the RTC-RTC autocorrelation. Also note that the 1.4Ghz galactic foreground emission that is visible in the last 30% of the measurement does not get flagged, even though it is of comparable amplitude as the lines of interference in the first 70%.



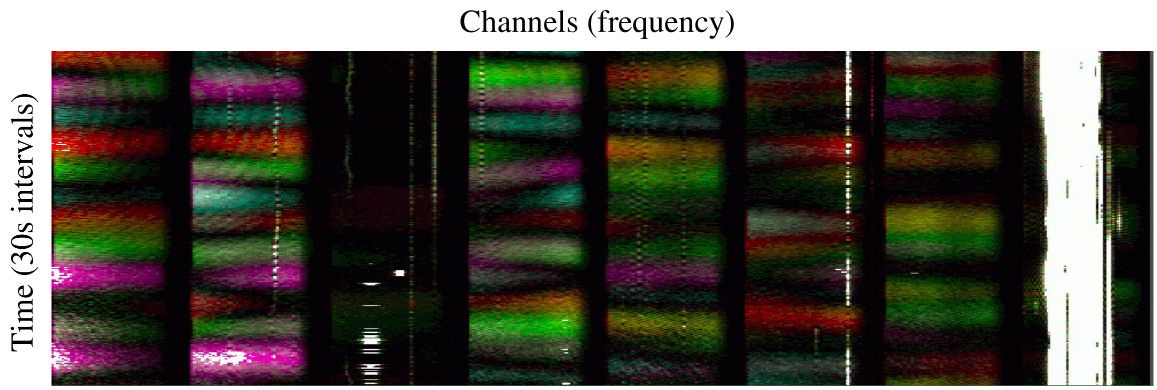
baseline RTC-RTD YY scale:0-0.002 (arbitrary units)

The output of the WSRT_flagger shown below again clearly shows that there is a problem with RTC, even if it was not able to flag all data:

```
15:32-375> WSRT_flagger ms=10406251_S0_T2.MS crosspol=false window=13 min=5 max=5 flagrms=true flagdata=true
Runnning flagger please wait...
WSRT_flagger: Version 0.41 WSRT flagging by Adriaan Renting and Michiel Brentjens 0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Spectral window: 1
       RT0    RT1    RT2    RT3    RT4    RT5    RT6    RT7    RT8    RT9    RTA    RTB    RTC    RTD
RT0    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT1    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT2    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT3    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT4    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT5    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT6    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT7    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT8    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RT9    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RTA    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RTB    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
RTC    27%    27%    27%    27%    27%    27%    27%    27%    27%    27%    27%    27%    51%    27%
RTD    0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     0%     27%    0%
Spectral windows (flagged %): SW0
                              3%
Antennae (flagged %):    RT0   RT1   RT2   RT3   RT4   RT5   RT6   RT7   RT8   RT9   RTA   RTB   RTC   RTD
                         1%    1%    1%    1%    1%    1%    1%    1%    1%    1%    1%    1%    28%   1%
Elapsed: 0:56.60   CPU: 43.064u+8.899s = 91.7%
```

4.3 Example 3

The third example is measurement 10403396, which is a LFFE low frequency measurement containing a lot of interference.

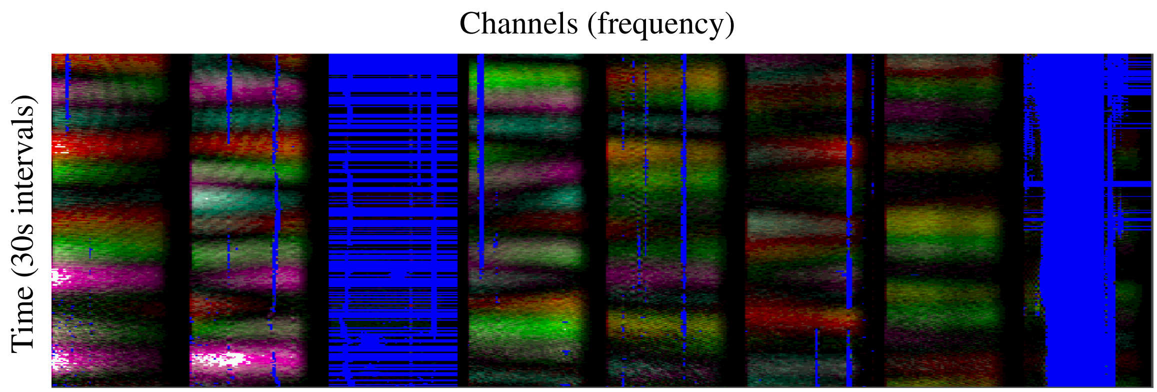


baseline RT0-RT3 YY scale:0-0.006 (arbitrary units)

To interpret these images correctly you need to know that the measurement contains 8 separate spectral windows. Because of the bandpass characteristics of the WSRT this results in 8 dark patches separating the bans of data when plotted together in one image. Amplitude is again plotted as intensity, and phase as color.

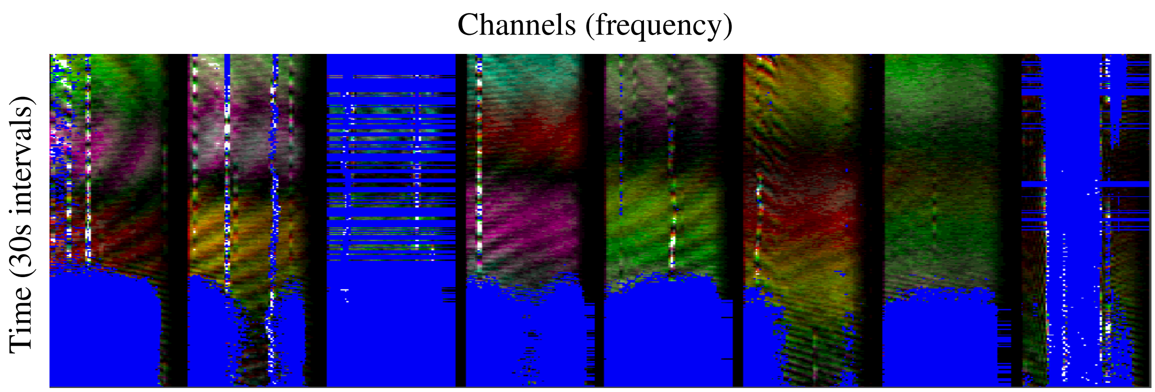
You can see in the image above that something seems to be wrong with spectral window 3 and that spectral window 8 contains a lot of interference. You can also see some interference in the other spectral windows.

The image below shows in blue which data point have been flagged using the WSRT_flagger. There is still some unflagged data in spectral window 3 that should probably be flagged, but otherwise most of the interference seems to be flagged. Most of the celestial signal remains, even where it's intensity is higher as some of the interference flagged, for example in the second spectral window.



baseline RT0-RT3XX scale:0-0.006 (arbitrary units)

The output of the WSRT_flagger for the 10403396 measurement is shown below. It is much more verbose, because it lists statistics for all spectral windows. From the output you can see that a lot of the data as short baselines is flagged. This is because at the end of the measurement there is some kind of interference, possibly the sun, that seems to affect especially the short baselines.



baseline RT0-RT1 YY scale:0-0.006 (arbitrary units)

```
17:05-400> WSRT_flagger ms=10403396_S0_T3.MS crosspol=false window=7 min=5 max=5 flagrms=true flagdata=true
Runnning flagger please wait...
WSRT_flagger: Version 0.41 WSRT flagging by Adriaan Renting and Michiel Brentjens 0% 10% 20% 30% 40% 50% 60%
70% 80% 90% 100%
Spectral window: 1
```

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	33%	19%	1%	0%	1%	0%	0%	0%	0%	0%	0%	3%	23%
RT1	33%	0%	41%	20%	8%	5%	0%	0%	0%	0%	0%	0%	2%	22%
RT2	19%	41%	0%	29%	11%	3%	1%	0%	0%	0%	0%	0%	1%	21%
RT3	1%	20%	29%	0%	29%	19%	6%	3%	0%	0%	0%	0%	0%	20%
RT4	0%	8%	11%	29%	0%	31%	16%	6%	0%	0%	0%	0%	0%	20%
RT5	1%	5%	3%	19%	31%	0%	39%	26%	7%	4%	0%	0%	0%	20%
RT6	0%	0%	1%	6%	16%	39%	0%	34%	17%	3%	1%	0%	0%	20%
RT7	0%	0%	0%	3%	6%	26%	34%	0%	32%	16%	10%	1%	0%	20%
RT8	0%	0%	0%	0%	0%	7%	17%	32%	0%	31%	20%	11%	0%	20%
RT9	0%	0%	0%	0%	0%	4%	3%	16%	31%	0%	6%	31%	0%	19%
RTA	0%	0%	0%	0%	0%	0%	1%	10%	20%	6%	0%	28%	0%	20%
RTB	0%	0%	0%	0%	0%	0%	0%	1%	11%	31%	28%	3%	0%	20%
RTC	3%	2%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	49%
RTD	23%	22%	21%	20%	20%	20%	20%	20%	20%	19%	20%	20%	49%	20%

Spectral window: 2

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	22%	13%	3%	3%	4%	2%	2%	1%	0%	0%	1%	5%	6%
RT1	22%	0%	28%	11%	7%	8%	6%	4%	0%	0%	0%	0%	3%	4%
RT2	13%	28%	0%	18%	5%	9%	9%	8%	0%	0%	0%	0%	3%	3%
RT3	3%	11%	18%	0%	17%	14%	21%	14%	3%	0%	0%	0%	1%	2%
RT4	3%	7%	5%	17%	0%	23%	26%	20%	10%	4%	2%	1%	0%	0%
RT5	4%	8%	9%	14%	23%	4%	30%	29%	15%	10%	6%	5%	0%	1%
RT6	2%	6%	9%	21%	26%	30%	0%	28%	20%	14%	10%	6%	0%	0%
RT7	2%	4%	8%	14%	20%	29%	28%	0%	26%	13%	8%	7%	0%	0%
RT8	1%	0%	0%	3%	10%	15%	20%	26%	0%	22%	8%	5%	0%	0%
RT9	0%	0%	0%	0%	4%	10%	14%	13%	22%	0%	2%	25%	0%	0%
RTA	0%	0%	0%	0%	2%	6%	10%	8%	8%	2%	0%	17%	0%	0%
RTB	1%	0%	0%	0%	1%	5%	6%	7%	5%	25%	17%	0%	1%	0%
RTC	5%	3%	3%	1%	0%	0%	0%	0%	0%	0%	0%	1%	0%	13%
RTD	6%	4%	3%	2%	0%	1%	0%	0%	0%	0%	0%	0%	13%	0%

Spectral window: 3

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	67%	78%	66%	67%	65%	66%	65%	65%	65%	65%	64%	65%	64%	64%
RT1	78%	9%	39%	38%	21%	19%	13%	12%	11%	11%	11%	10%	9%	9%
RT2	66%	39%	9%	31%	21%	13%	13%	11%	11%	11%	10%	11%	8%	8%
RT3	67%	38%	31%	8%	34%	30%	25%	13%	11%	11%	11%	10%	8%	8%
RT4	65%	21%	21%	34%	8%	36%	50%	21%	10%	10%	8%	9%	7%	7%
RT5	66%	19%	13%	30%	36%	9%	43%	45%	17%	12%	11%	11%	9%	8%
RT6	65%	13%	13%	25%	50%	43%	8%	38%	34%	13%	11%	11%	9%	8%
RT7	65%	12%	11%	13%	21%	45%	38%	8%	37%	29%	20%	13%	9%	9%
RT8	65%	11%	11%	11%	10%	17%	34%	37%	8%	36%	25%	31%	9%	9%
RT9	65%	11%	11%	11%	10%	12%	13%	29%	36%	8%	20%	39%	10%	9%
RTA	64%	11%	10%	11%	8%	11%	11%	20%	25%	20%	8%	37%	8%	8%
RTB	65%	10%	11%	10%	9%	11%	11%	13%	31%	39%	37%	9%	9%	9%
RTC	64%	9%	8%	8%	7%	9%	9%	9%	9%	10%	8%	9%	7%	35%
RTD	64%	9%	8%	8%	7%	8%	8%	9%	9%	9%	8%	9%	35%	7%

Spectral window: 4

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	29%	5%	2%	2%	2%	2%	2%	0%	0%	0%	0%	2%	3%
RT1	29%	0%	29%	7%	3%	3%	3%	2%	0%	0%	0%	0%	1%	1%
RT2	5%	29%	0%	28%	6%	4%	3%	3%	0%	0%	0%	0%	0%	0%
RT3	2%	7%	28%	0%	31%	11%	11%	8%	4%	3%	1%	1%	0%	1%
RT4	2%	3%	6%	31%	0%	35%	15%	11%	4%	3%	2%	1%	0%	0%
RT5	2%	3%	4%	11%	35%	0%	33%	22%	8%	8%	5%	4%	0%	0%
RT6	2%	3%	3%	11%	15%	33%	0%	32%	12%	10%	6%	6%	0%	0%
RT7	2%	2%	3%	8%	11%	22%	32%	0%	34%	11%	6%	5%	0%	0%
RT8	0%	0%	0%	4%	4%	8%	12%	34%	0%	32%	10%	12%	0%	0%
RT9	0%	0%	0%	3%	3%	8%	10%	11%	32%	0%	4%	37%	0%	0%
RTA	0%	0%	0%	1%	2%	5%	6%	6%	10%	4%	0%	30%	0%	0%
RTB	0%	0%	0%	1%	1%	4%	6%	5%	12%	37%	30%	0%	0%	0%
RTC	2%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	32%
RTD	3%	1%	0%	1%	0%	0%	0%	0%	0%	0%	0%	0%	32%	0%

Spectral window: 5

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	31%	18%	4%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
RT1	31%	0%	36%	21%	2%	1%	5%	1%	0%	0%	0%	0%	0%	0%
RT2	18%	36%	0%	33%	15%	3%	6%	4%	1%	0%	0%	1%	0%	0%
RT3	4%	21%	33%	1%	30%	12%	7%	5%	3%	1%	0%	0%	0%	0%
RT4	0%	2%	15%	30%	0%	32%	18%	2%	0%	0%	0%	0%	0%	0%
RT5	0%	1%	3%	12%	32%	0%	35%	20%	1%	1%	0%	0%	0%	0%
RT6	0%	5%	6%	7%	18%	35%	0%	36%	18%	3%	0%	0%	0%	0%
RT7	0%	1%	4%	5%	2%	20%	36%	0%	34%	14%	5%	0%	0%	0%
RT8	0%	0%	1%	3%	0%	1%	18%	34%	0%	33%	16%	24%	0%	0%
RT9	0%	0%	0%	1%	0%	1%	3%	14%	33%	0%	9%	38%	0%	0%
RTA	0%	0%	0%	0%	0%	0%	0%	5%	16%	9%	0%	35%	0%	0%
RTB	0%	0%	1%	0%	0%	0%	0%	0%	24%	38%	35%	0%	0%	0%
RTC	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	33%
RTD	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	33%	0%

Spectral window: 6

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	9%	1%	4%	0%	2%	0%	0%	1%	0%	0%	0%	0%	0%
RT1	9%	0%	20%	18%	2%	4%	1%	0%	1%	3%	0%	0%	0%	0%
RT2	1%	20%	0%	10%	0%	1%	1%	1%	1%	0%	0%	0%	0%	0%
RT3	4%	18%	10%	1%	9%	5%	4%	1%	5%	4%	1%	0%	0%	0%
RT4	0%	2%	0%	9%	0%	10%	2%	2%	2%	0%	0%	0%	0%	0%
RT5	2%	4%	1%	5%	10%	0%	10%	3%	4%	5%	1%	0%	0%	0%
RT6	0%	1%	1%	4%	2%	10%	0%	12%	2%	2%	0%	1%	0%	0%
RT7	0%	0%	1%	1%	2%	3%	12%	0%	12%	1%	0%	0%	0%	0%
RT8	1%	1%	1%	5%	2%	4%	2%	12%	0%	10%	2%	3%	0%	1%
RT9	0%	3%	0%	4%	0%	5%	2%	1%	10%	0%	1%	18%	0%	0%
RTA	0%	0%	0%	1%	0%	1%	0%	0%	2%	1%	0%	14%	0%	0%
RTB	0%	0%	0%	0%	0%	0%	1%	0%	3%	18%	14%	0%	0%	0%
RTC	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	13%
RTD	0%	0%	0%	0%	0%	0%	0%	0%	1%	0%	0%	0%	13%	0%

Spectral window: 7

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	0%	26%	4%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
RT1	26%	0%	27%	8%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
RT2	4%	27%	0%	21%	2%	1%	1%	0%	0%	3%	0%	0%	0%	0%
RT3	0%	8%	21%	0%	20%	3%	2%	0%	0%	1%	0%	0%	0%	0%
RT4	0%	0%	2%	20%	0%	22%	5%	1%	0%	0%	0%	0%	0%	0%
RT5	0%	0%	1%	3%	22%	0%	30%	8%	1%	0%	0%	0%	0%	0%
RT6	0%	0%	1%	2%	5%	30%	0%	27%	7%	2%	0%	0%	0%	0%
RT7	0%	0%	0%	0%	1%	8%	27%	0%	26%	4%	1%	0%	0%	0%
RT8	0%	0%	0%	0%	0%	1%	7%	26%	0%	25%	7%	11%	0%	0%
RT9	0%	0%	3%	1%	0%	0%	2%	4%	25%	0%	5%	33%	0%	0%
RTA	0%	0%	0%	0%	0%	0%	0%	1%	7%	5%	0%	31%	0%	0%
RTB	0%	0%	0%	0%	0%	0%	0%	0%	11%	33%	31%	0%	0%	0%
RTC	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	32%
RTD	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	32%	0%

Spectral window: 8

	RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
RT0	18%	54%	62%	63%	65%	68%	75%	62%	73%	67%	72%	57%	64%	66%
RT1	54%	5%	49%	55%	62%	66%	72%	54%	69%	68%	67%	52%	59%	61%
RT2	62%	49%	13%	51%	63%	74%	77%	56%	70%	70%	74%	57%	60%	61%
RT3	63%	55%	51%	7%	58%	70%	77%	57%	68%	63%	71%	57%	57%	59%
RT4	65%	62%	63%	58%	20%	73%	80%	65%	76%	68%	74%	61%	65%	65%
RT5	68%	66%	74%	70%	73%	31%	82%	71%	83%	79%	82%	69%	76%	76%
RT6	75%	72%	77%	77%	80%	82%	44%	73%	85%	87%	87%	79%	84%	82%
RT7	62%	54%	56%	57%	65%	71%	73%	3%	70%	68%	77%	64%	73%	74%
RT8	73%	69%	70%	68%	76%	83%	85%	70%	35%	78%	82%	73%	84%	85%
RT9	67%	68%	70%	63%	68%	79%	87%	68%	78%	28%	64%	63%	79%	80%
RTA	72%	67%	74%	71%	74%	82%	87%	77%	82%	64%	36%	61%	85%	86%
RTB	57%	52%	57%	57%	61%	69%	79%	64%	73%	63%	61%	9%	77%	77%
RTC	64%	59%	60%	57%	65%	76%	84%	73%	84%	79%	85%	77%	36%	81%
RTD	66%	61%	61%	59%	65%	76%	82%	74%	85%	80%	86%	77%	81%	42%

Spectral windows (flagged %):

SW0	SW1	SW2	SW3	SW4	SW5	SW6	SW7
9%	6%	23%	6%	6%	2%	4%	66%

Antennae (flagged %):

RT0	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RTA	RTB	RTC	RTD
18%	14%	14%	15%	15%	17%	18%	16%	17%	15%	14%	14%	12%	14%

Elapsed: 2:02.86 CPU: 97.680u+11.633s = 88.9%

5 Appendix 1: Data description

The Flagger needs several numbers, which is calculates from datafields available in the Measurement Set in the way listed below:

Number of Samples = numbers of rows of the MS MAIN table.
Number of Fields = numbers of rows of the MS FIELD table.
Number of Antennae = numbers of rows of the MS ANTENNA table.
Antenna Names = strings from the MS ANTENNA NAME column.
Number of Channels = number from the first cell in the NUM_CHAN column in the MS SPECTRAL_WINDOW table.
NumPolarizations = number from the first cell in the NUM_CORR column in the MS POLARIZATION table.
Exposure Time = number from the first cell of the EXPOSURE column in the MS MAIN table.
Channel Bandwidth = number from the first cell of the TOTAL_BANDWIDTH column in the MS SPECTRAL_WINDOW table divided by the Number of Channels.
Estimated Minimal Noise = one over the square root of Exposure Time times Channel Bandwidth.
Interval = number from the first cell of the INTERVAL column in the MS MAIN table.
Number of Timeslots = number from the last cell minus the number from the first cell of the TIME_CENTROID column in the MS MAIN table, divided by the Interval, plus one.
Number of Baselines = Number of Antennae times Number of Antennae plus one, divided by two.
NumBands = the Number of Samples, divided by the Number of Baselines times the Number of Timeslots.
Data to flag = complex numbers from either the DATA, CORRECTED_DATA or MODEL_DATA column in the MS MAIN table, and the related numbers that describe it from ANTENNA1, ANTENNA2, DATA_DESC_ID, FIELD_ID, TIME_CENTROID.
Flag values = boolean values from the FLAGS column in the MS MAIN table.