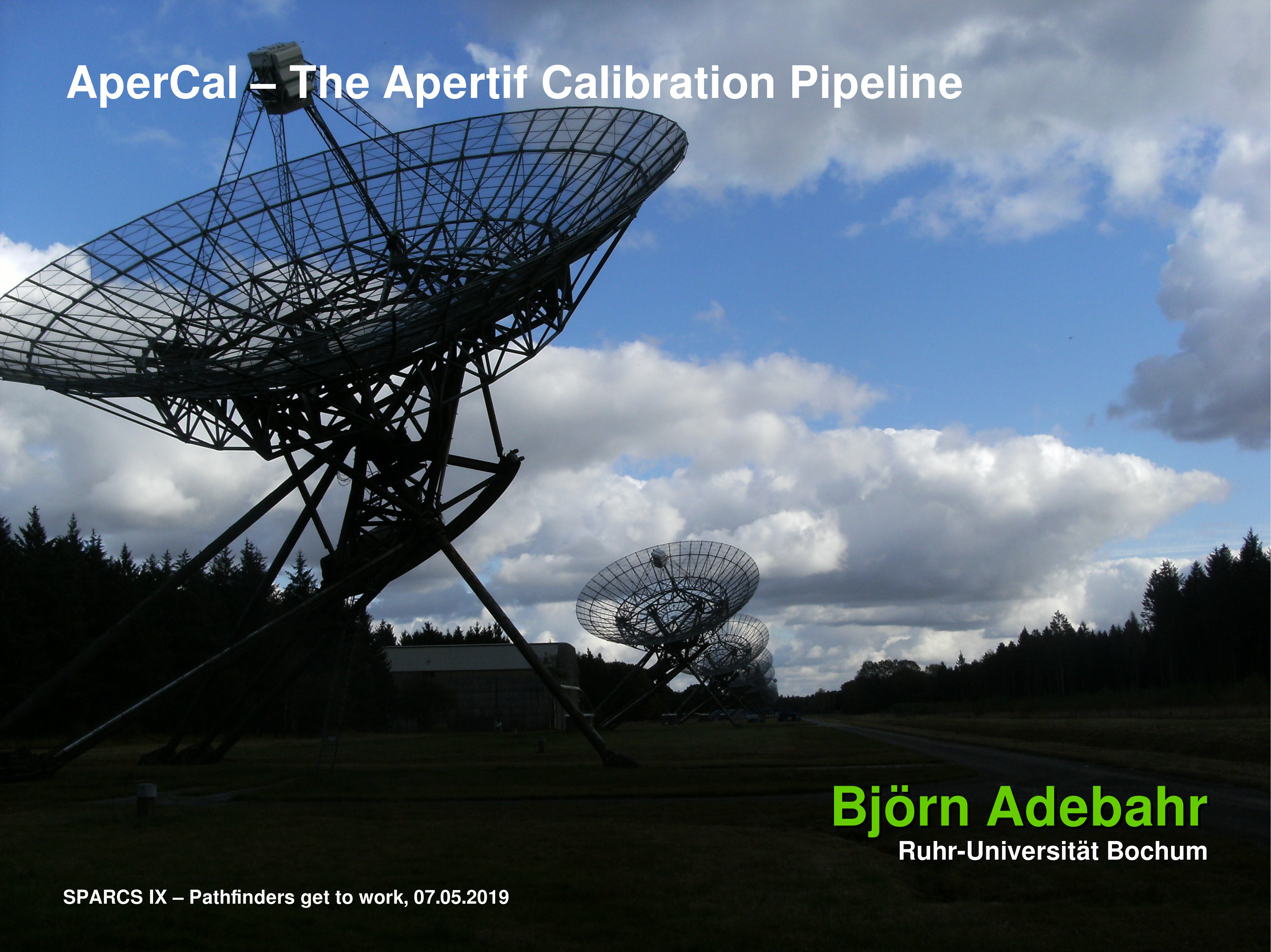
AperCal – The Apertif Calibration Pipeline

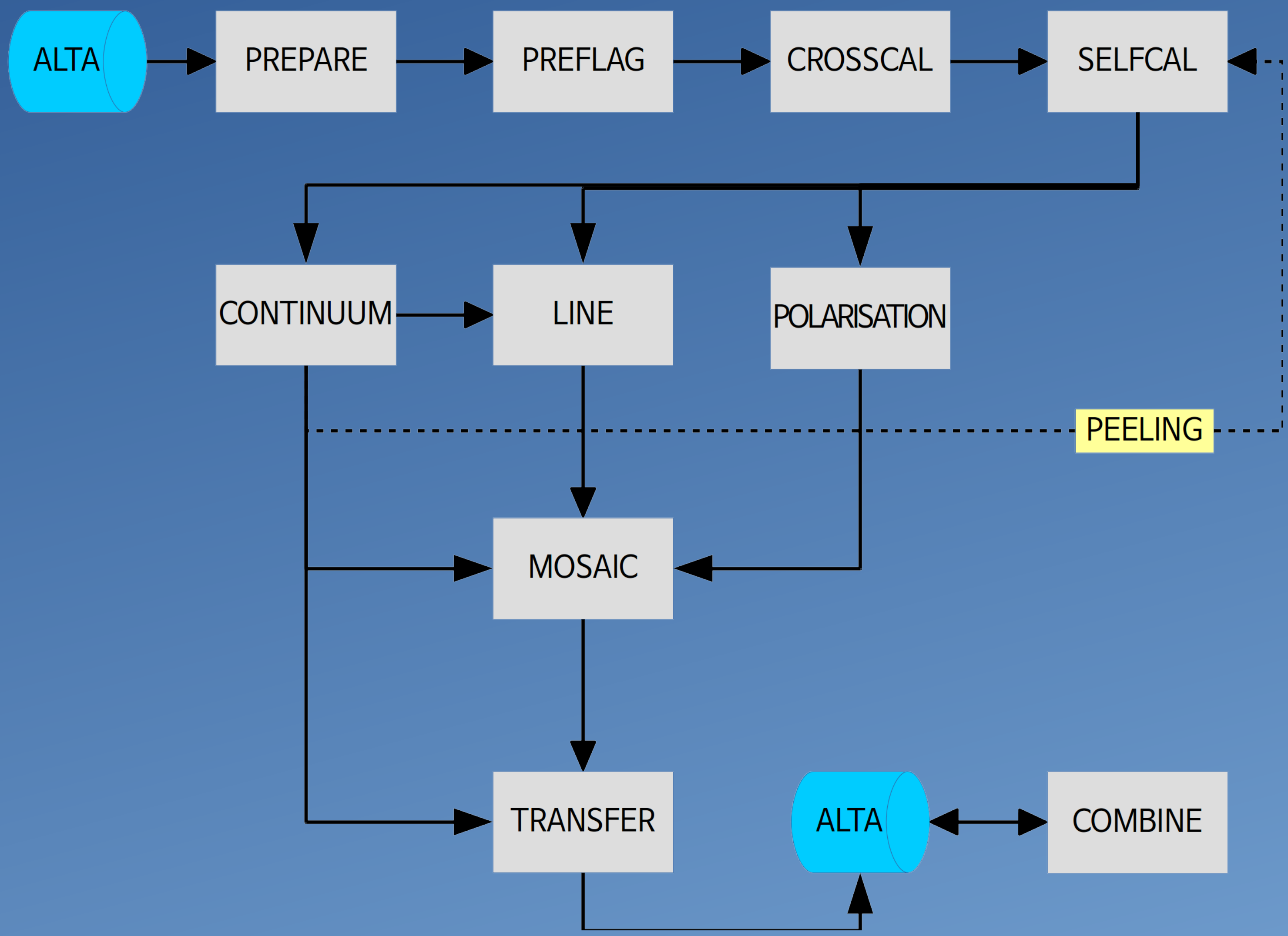


Björn Adebahr
Ruhr-Universität Bochum

SPARCS IX – Pathfinders get to work, 07.05.2019

AperCal structure

- CASA and MIRIAD routines using a jupyter notebook framework
- Calibration/Imaging steps are independent software routines (modules)
- All 40 beams are processed independently and in parallel



AperCal output data products

APERTIF output products are arranged in levels depending on the needed effort to create them

Level 0

- Raw uncalibrated visibilities

} Data from the telescope

Level 1

- Calibrated visibilities for each beam

Level 2

- Continuum images of individual beams and mosaics
- Line cubes with full frequency resolution
- Stokes Q and U cubes with ~0.8 MHz resolution
- Stokes V images
- Spectral index maps

} Pipeline reduction (Levels 1–2)

Level 3

- Sub-cubes for line sources
- Moment 0 and 1 maps
- RM-cubes
- Electric field vector images (PA)
- Magnetic field vector images (PA0)
- Polarised intensity images (linear and circular)

} Reduction within the pipeline framework

AperCal automatic processing

- Config file for each observation and beam with standard parameters
- Future development: Find optimal parameters depending on type of observed field

jupyter 2 - Calibration and imaging of several APERTIF element beams Last Checkpoint: 04/18/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Notebook saved | Trusted | Python 2

Run | Code

First we reduce all beams with the same standard paramaters.

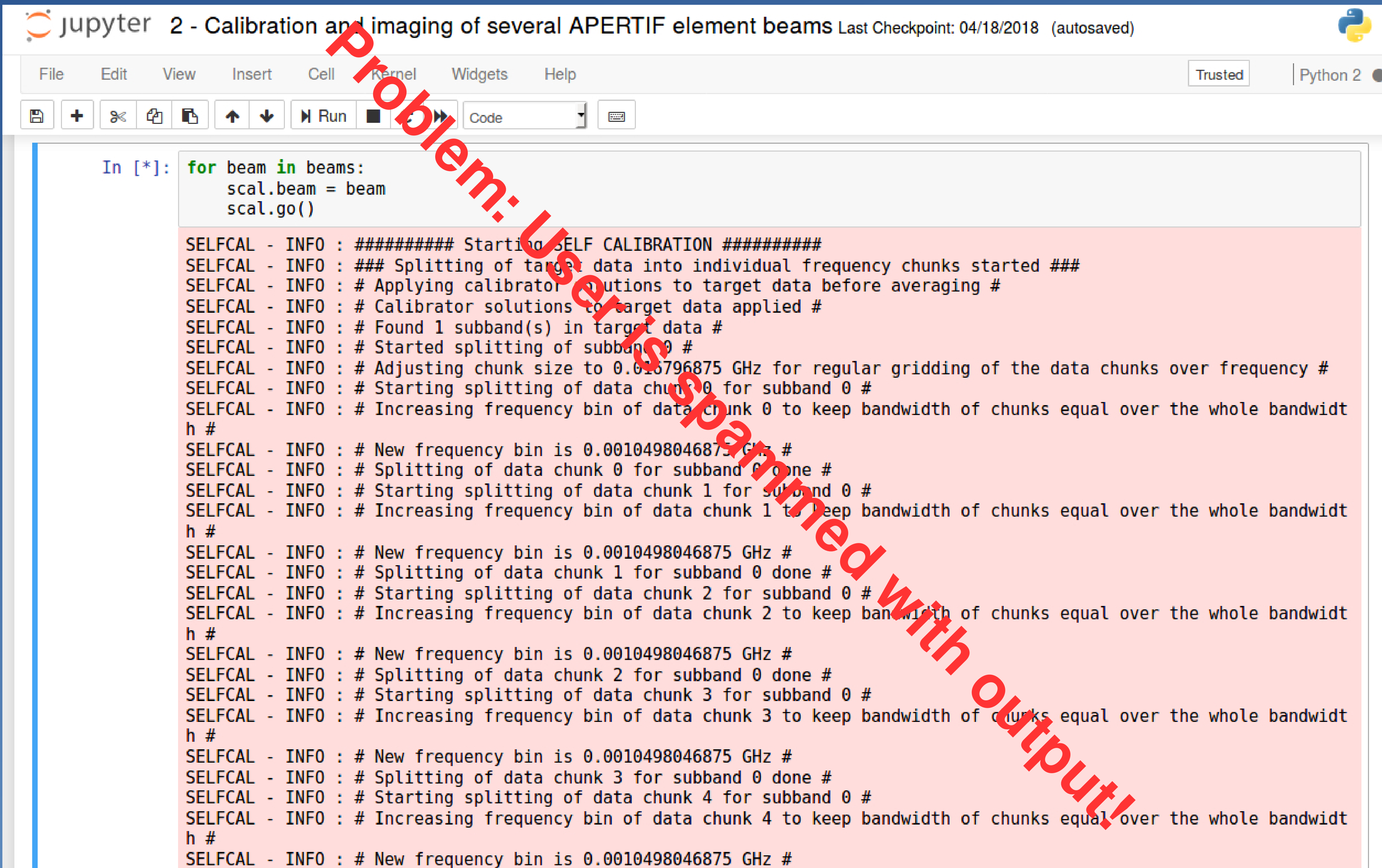
In [2]:
```
scal = apercal.scal('/home/{}/apercal/ipython-notebooks/tutorials/cfg/2.cfg'.format(myusername))
scal.show()
```

```
SELFCAL - INFO : ### Configuration file /home/adebahr/apercal/ipython-notebooks/tutorials/cfg/2.cfg successfully read
! ###
```

```
SELFCAL
        selfcal_image_imsize = 2049
        selfcal_image_cellsize = 4
        selfcal_refant =
        selfcal_splitdata = True
        selfcal_splitdata_chunkbandwidth = 0.02
        selfcal_splitdata_channelbandwidth = 0.001
        selfcal_flagantenna =
        selfcal_flagline = True
        selfcal_flagline_sigma = 0.5
        selfcal_parametric = True
        selfcal_parametric_skymodel_radius = 0.5
        selfcal_parametric_skymodel_cutoff = 0.8
        selfcal_parametric_skymodel_distance = 30
        selfcal_parametric_solint = 5
        selfcal_parametric_uvmin = 0.5
        selfcal_parametric_uvmax = 1000
        selfcal_parametric_amp = False
        selfcal_standard_majorcycle = 3
        selfcal_standard_majorcycle_function = square
        selfcal_standard_minorcycle = 3
        selfcal_standard_minorcycle_function = square
        selfcal_standard_c0 = 10.0
        selfcal_standard_c1 = 5.0
        selfcal_standard_minorcycle0_dr = 5.0
        selfcal_standard_drinit = 50
```

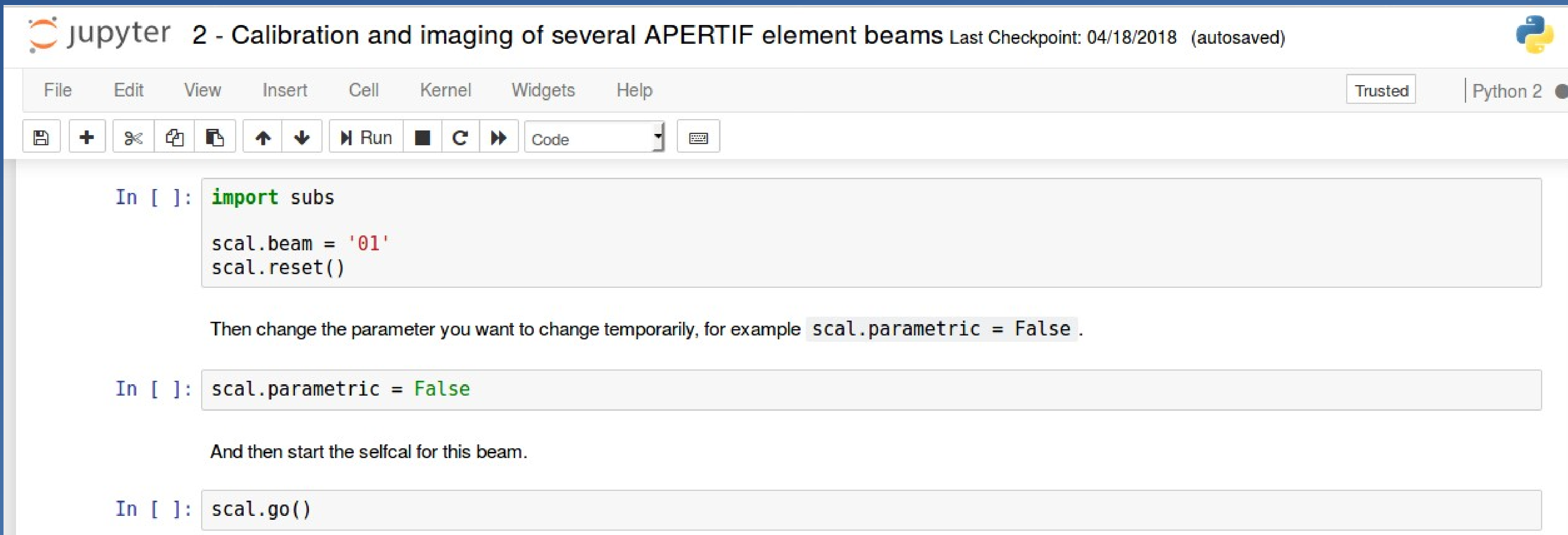
AperCal logger output

- All actions are logged and saved to a file
- Different logger levels



AperCal user interaction

- Parameters can be temporarily changed by the user in the interface
- We will never reach a 100% success rate for the calibration



Keep the complex information the pipeline produces

BUT

Summarise the information for the user so that (s)he can find the problem

AperCal summaries

- Summaries intend to provide the user with the neccesary information to find problems
- Reads the complex information from the pipeline reduction and shows at which step the calibration/imaging failed

In [4]:
```
ds = mosaic.detailed_summary_continuumstacked()
ds.style

d = dict(selector="th", props=[('text-align', 'center')])
ds.style.set_properties(**{'text-align':'center'})
```

Out[4]:

	Continuum calibration?	Copy successful?	Bmaj ["]	Bmin ["]	Bpa [deg]	Beam parameters?	Convol successful?	Weight	Residual RMS	Image accepted?
00	True	True	18.28	14.02	-0.83	False	False	nan	nan	False
01	True	True	18.48	14.37	0.8	True	True	1.04435	0.000253681	True
02	False	False	nan	nan	nan	False	False	nan	nan	False
03	False	False	nan	nan	nan	False	False	nan	nan	False
04	False	False	nan	nan	nan	False	False	nan	nan	False
05	False	False	nan	nan	nan	False	False	nan	nan	False
06	False	False	nan	nan	nan	False	False	nan	nan	False
07	False	False	nan	nan	nan	False	False	nan	nan	False
08	False	False	nan	nan	nan	False	False	nan	nan	False
09	False	False	nan	nan	nan	False	False	nan	nan	False
10	True	True	18.29	14.02	-0.96	True	True	0.955651	0.000265192	True
11	False	False	nan	nan	nan	False	False	nan	nan	False
12	False	False	nan	nan	nan	False	False	nan	nan	False
13	False	False	nan	nan	nan	False	False	nan	nan	False
14	False	False	nan	nan	nan	False	False	nan	nan	False
15	False	False	nan	nan	nan	False	False	nan	nan	False
16	False	False	nan	nan	nan	False	False	nan	nan	False
17	False	False	nan	nan	nan	False	False	nan	nan	False

Metrics

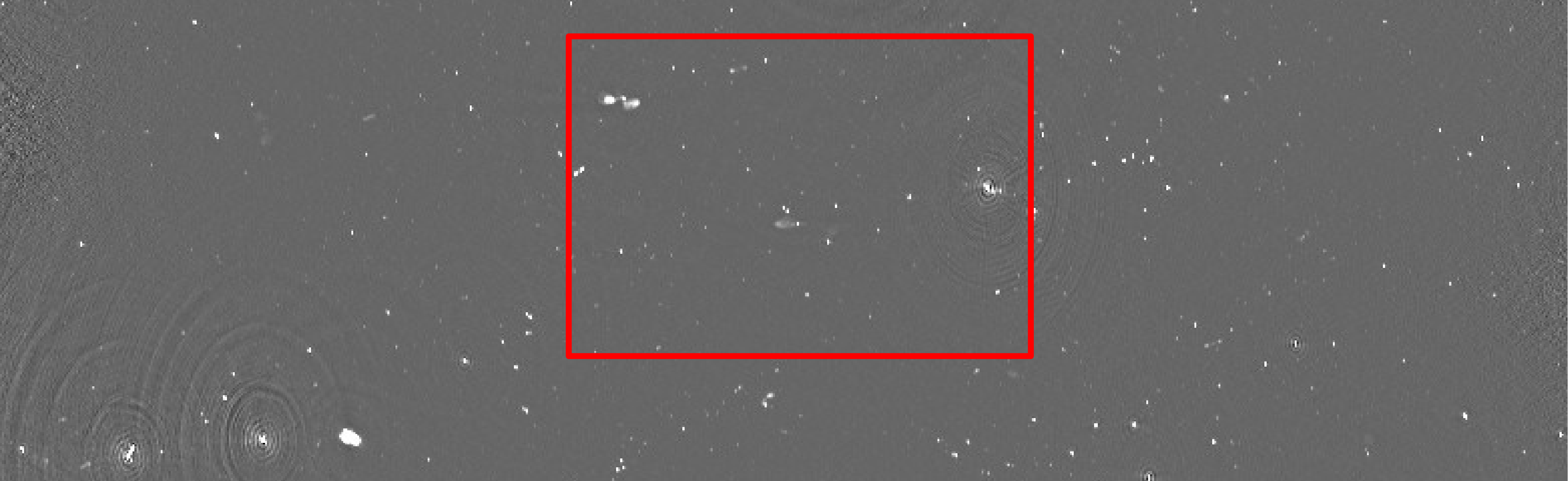
- Have to validate a calibration step or data product
- Stabilise the pipeline runs
- Only two values possible (a computer only understands True or False)
- Exact values often specific to a certain system/telescope

Some examples for selfcal/continuum imaging (implemented in AperCal):

- Is the rms of the final residual images unrealistically high?
- Are the statistics of the final residual images Gaussian?
- Is the cleaning mask occupying a large percentage of the image?
- Do the clean components show unrealistic high or low values?
- Does the summarised flux of the clean components after amplitude calibration significantly differ from the previous phase only calibration?
- If one of the above is flagged as bad use the solutions/images from the previous cycle

AperCal multi-frequency continuum image of S2246+38

- All 40 beams calibrated successfully
- 300 MHz bandwidth (220 MHz effectively)
- Created without human interaction within 24 hours of processing time
- Noise ~40 μJy/beam

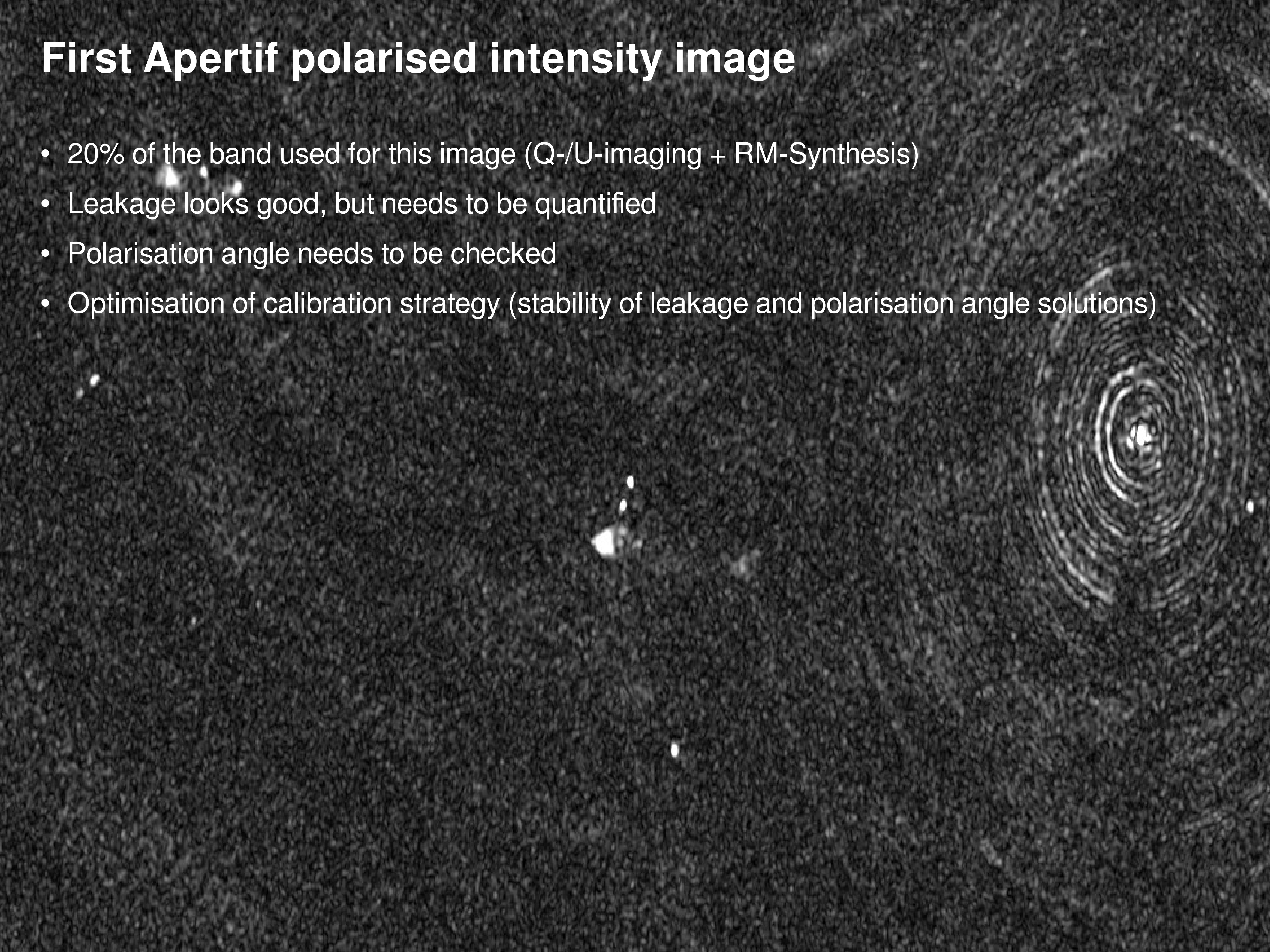


Improvements possible with

1. Amplitude calibration
2. Only use RFI free areas for calibration and imaging

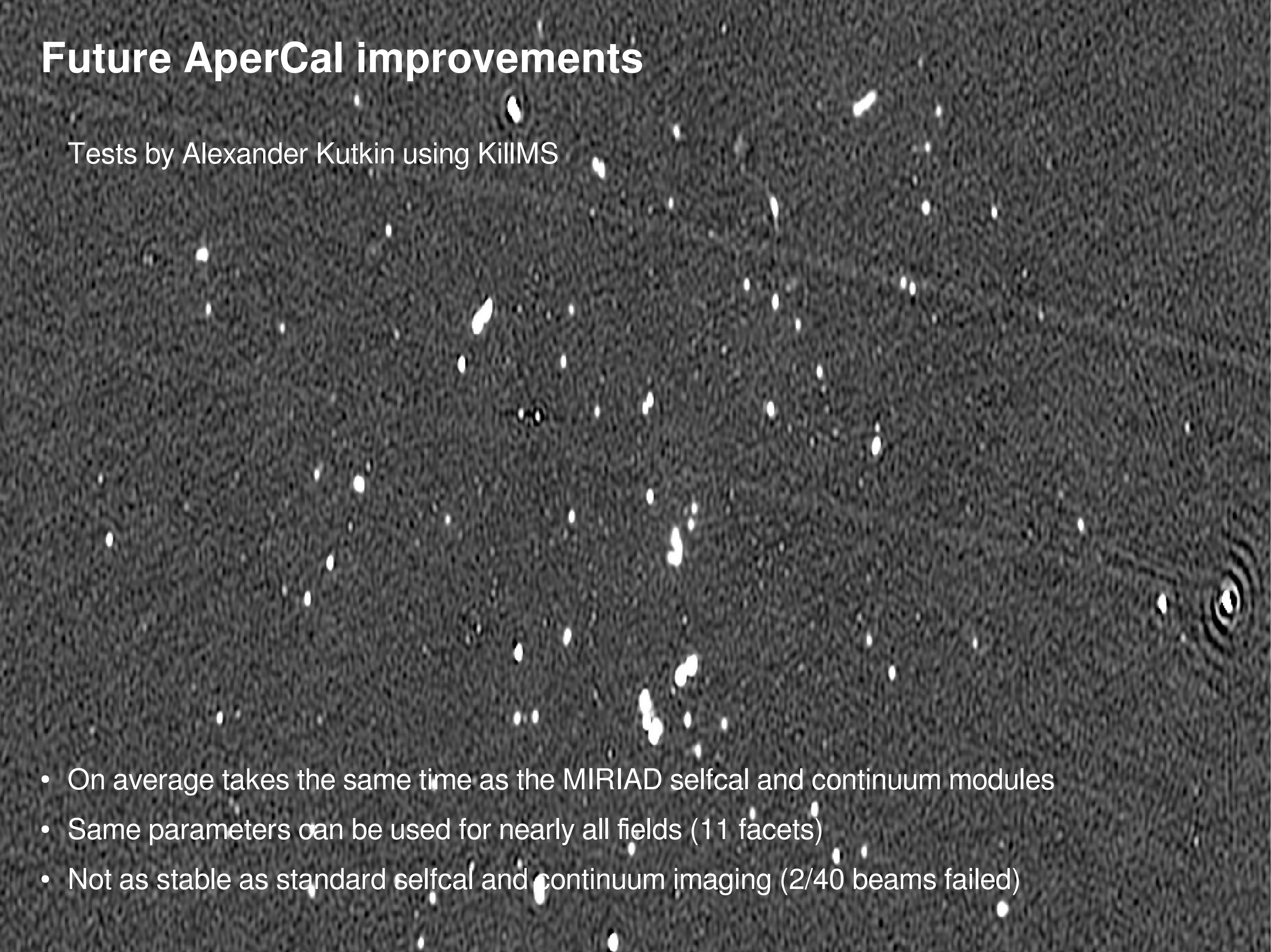
First Apertif polarised intensity image

- 20% of the band used for this image (Q-/U-imaging + RM-Synthesis)
- Leakage looks good, but needs to be quantified
- Polarisation angle needs to be checked
- Optimisation of calibration strategy (stability of leakage and polarisation angle solutions)



Future AperCal improvements

Tests by Alexander Kutkin using KillMS



- On average takes the same time as the MIRIAD selfcal and continuum modules
- Same parameters can be used for nearly all fields (11 facets)
- Not as stable as standard selfcal and continuum imaging (2/40 beams failed)

Future AperCal improvements

Short term (until surveys start in July)

- Option to split out a part of the band for a Quicklook pipeline and reduce data size
- Optimising pybdsf masking parameters
- Implement primary beam shape into parametric selfcal and mosaicking module

Long term (after surveys start)

- Dynamic range improvements (Peeling/DD-calibration, different parameters)
- Fully cleaned line cubes
- Data analysis framework
  - Continuum, polarisation and HI source finding
  - Automatic catalogue creation
  - Faraday cube reduction (PI-, PA-, RM-maps)