LOFAR GSM database project

Alexey Mints

February 19, 2013

1 Those involved:

1. Ger van Diepen [diepen at astron.nl];
2. George Heald [heald at astron.nl];
3. **Alexey Mints** [a.mints at hs.uni-hamburg.de];
4. Ronald Nijboer [rnijboer at astron.nl];
5. Bart Scheers [bartscheers at gmail.com];

2 Abbreviations

The following abbreviations and designations are used in this document as well as in the code.

GSM — Global Sky Model;

KSP — Key Science Project (of the LOFAR);

MSSS — Multifrequency Snapshot Sky Survey;

ra — right ascension;

decl — declination;

pa — positional angle;

fov — field of view;

Note 1: if not stated otherwise, by *band* we mean a combination of the frequency band and Stokes parameter.

Note 2: by *size* of the extended source we mean Gaussian parameters: minor and major axes and positional angle.

Note 3: by *detections* we will mean newly detected objects, reported by the source-finder and passed to the GSM, and by *sources* we will mean objects already stored in the database.

3 Project description

3.1 Goals

Main goal of the project is to provide a database solution for the Global Sky Model (hereafter GSM), basing on MSSS data [6, 2, 3]. For a given field of view BBS-format file with the list of the known source should be produced.

The pipeline is based on the one developed for the Transient KSP [5].

3.2 Software requirements

The next programs/modules are needed to run the GSM pipeline:

MonetDB or PostgreSQL ;

Python tested on versions 2.6 and 2.7;

psycopg2 if PostgreSQL is used;

numpy ;

configobj or lofar.parameterset ;

healpy (includes HealPix library);

To check if all requirements are met one may use `validate_install.py` script, provided with the package (see 7.5.2).

4 Inputs

4.1 Input data flow

Data comes from pyBDSM or pySE source detection tools in the MSSS pipeline. Inputs for every step are:

a Parset file, containing image properties and list of references to the catalog files;

b Catalog files that are ASCII-tables produced by a source-finder.

Data is first passed to the `detections` table[1]. Then it is copied (with proper transformations and filtering) into a `extractedsources` table. Then matching to known sources is done and results are stored in the `temp_associations` table. Source positions/fluxes are updated or inserted into `runningcatalog` and `runningcatalog_fluxes` as needed according to the `temp_associations` table. Relevant associations between `extractedsources` and `runningcatalog` are kept in `assocxtrsources`.

5 Outputs

5.1 Tables and columns

5.1.1 Basic information

At the end data is loaded into the database into several tables [2]:

1. `images` — one record describing the image (unique ID, frequency, fov, pointing etc.);

2. `image_stats` — statistics on the image associations;

3. `extractedsources` — all sources detected in each image;

4. `temp_assocxtrsources` — temporary associations between detections in `extractedsources` and sources from `runningcatalog`. All possible associations are inserted here for each image. Emptied after all associations will be processed and all problems resolved;

5. `assocxtrsources` — actual associations between detections in `extractedsources` and sources from `runningcatalog`;

6. `runningcatalog` — positions and sizes of sources included into the catalog;

7. `runningcatalog_fluxes` — fluxes (in each band) of sources from runningcatalog;

[1] The full list of tables with descriptions see below.

[2] For a detailed database description see 7.2

Number of cross-matches to `extractedsources` is stored as well in the `runningcatalog` and `runningcatalog_fluxes` tables in `datapoints` colunms.

For each value V (can be: ra, decl, major/minor axes, positional angle, peak/total flux) several columns are used in `runningcatalog` or `runningcatalog_fluxes` tables. Weights are assigned given measured error ΔV as $w = \frac{1}{(\Delta \mathrm{V})^2}$. Columns for each V are:

wm_V — weighted average value $\frac{\sum w_i \mathrm{V}_i}{\sum w_i}$;

wm_V_err — weighted error $\sqrt{\frac{1}{\sum w_i}}$;

avg_wV — sum of weighted values $\sum w_i \mathrm{V}_i$;

avg_weight_V — sum of weights $\sum w_i$;

5.1.2 Storing and matching extended sources

Extended sources are stored in a special way. As they can look different in different frequencies, we have to store positional and size information in the `runningcatalog` table in a per-band way. One extra record without band is stored as well, and named “Cross-band” source. For a cross-band source Gaussian parameters are weighted sums from per-band parameters for this source. For each record in the `extractedsources` table there are therefore two records in `assocxtrsources` table — one for the cross-band record in the `runningcatalog` and one for the per-band record.

For a physical reason it might also happen, that some extended source will have more than one sub-component in a given band. Thus it is allowed for an extended source to contain several per-band sources bound to one cross-band source. Note that in the general case a sum of `datapoints` values for per-band sources is *greater or equal* then `datapoints` value for a cross-band source (see section 6.5.1). Also there might arise a problem that two or more extended sources share the same cross-band source.

5.2 Formulas

Here we introduce a special notation used to describe problem resolution:

A-D — point sources;

K-Q — extended sources (cross-band);

k-q — extended sources (per-band);

X-Z — detections (always single-band);

0-9 — band (might be multiple comma-separated values);

() — weighted average;

With such a notation a problem resolution might be expressed as a transformation from one source set to another.

Examples:

$$
\begin{aligned}
A1 + X1 &= (AX)1 && (1)\\
A1 + X2 &= (AX)1,2 && (2)\\
Kk1,l2 + X1 &= (KX)(kX)1,l2 && (3)\\
Kk1,l2 + X3 &= (KX)k1,l2,m3 && (4)\\
A1,2,3 + X1 + Y1 &= (AX)1,2,3 + (AY)1,2,3 && (5)\\
A1,2,3 + B1,2,3 + X1 &= (AX)1,2,3 + B1,2,3 && (6)\\
Kk1,l2 + X1 + Y1 &= (KX)(kX)1,l2 + (KY)(kY)1,l2 && (7)\\
Kk1,l2 + Lk1,l2 + X1 &= (KX)(kX)1,l2 + Lk1,l2 && (8)\\
Kk1,l2 + Lk1,l2 + X3 &= (KX)k1,l2,m3 + Lk1,l2 && (9)\\
Kk1,l2 + Lm3,n4 + X5 &= (KLX)k1,l2,m3,n4,o5 && (10)\\
A1 + B1 + X1 + Y1 &= \text{Group1} && (11)
\end{aligned}
$$

For example, in equation 7 means, that we have matched two detections (X and Y) in band 1 to one extended (cross-band) source K that has per-band sources k in band 1 and l in band 2. As a result we get two cross-band sources:

1. source with cross-band position equal to weighted average of positions of sources K and X, position and flux in band 1 equal to weighted average of positions and flux of sources k and X, position and flux in band 2 equal to position and flux of source l;

2. source with cross-band position equal to weighted average of positions of sources K and Y, position and flux in band 1 equal to weighted average of positions and flux of sources k and Y, position and flux in band 2 equal to position and flux of source l;

5.3 API

ToDo.

5.4 Errors

If error an is detected, it is reported to output and to the log. Whenever possible, the transaction is rolled back.

6 Databases and pipelines

6.1 Matching criteria

Sources for matching are pre-selected basing on the following conditions:

- source kind (point-like or extended) has to be the same;
- angular distance between source centers has to be below r_{min}, set in the settings file, see 6.2;

Matching criteria for point sources is based on de Ruiter distance [1]:

$$R^2_{deRuiter} = \frac{(\alpha_1 \cos\delta_1 - \alpha_2 \cos\delta_2)^2}{(\Delta\alpha_1)^2 + (\Delta\alpha_2)^2} + \frac{(\delta_1 - \delta_2)^2}{(\Delta\delta_1)^2 + (\Delta\delta_2)^2} \quad (12)$$

where $\alpha_{1,2}$ and $\delta_{1,2}$ are right ascensions and declinations of the two sources and $\Delta\alpha$ and $\Delta\delta$ are reported positional errors.

The first point comes from the `runningcatalog` and the second one – from `extractedsources`. For all pairs that have $R_{deRuiter} < R_x$, where R_x is a chosen critical value, an association is possible. For such a pair a record is inserted into `temp_runningcatalog`, containing references to both objects, $R_{deRuiter}$ and distance (in arcsec) $r_{arc} = 2\arcsin(r_{12}/2)$ where $r_{12} = \sqrt{(r_{1,x} - r_{2,x})^2 + (r_{1,y} - r_{2,y})^2 + (r_{1,z} - r_{2,z})^2}$ is the 3D geometrical distance between sources.

For extended sources an alternative formula is used:

$$\tilde{R}^2_{deRuiter} = \frac{r^2_{arc}}{g^2_{major,1} + g^2_{major,2}}, \quad (13)$$

where g_{major} is a reported major axis. The criteria is then $\tilde{R}_{deRuiter} < \tilde{R}_x$.

6.2 Matching settings file

A special file named `settings.ini` is located at the src folder. It contains several settings:

Value	Parameter name	Default
R_x	match_distance	1.0
$\tilde{R}_x$	match_distance_extended	0.5
r_{min}	maximum_association_distance	0.05 (radian)
matcher type	matcher	- (should be SQL or F90)

6.3 Matching across 360 degrees border

It might happen, that source and detection will be close by, but separated by zero point in right ascension. I.e. the source will have $ra = 0°.001$ and the detection $ra = 359°.99$. They have to match, but eq. 12 will fail. To override this issue for an detection close to the 0-degree border ($|\alpha - 360°| < (\cos\delta)^{-1}$), a copy is created, mirrored along this border. A copied detection carries the id of the original.

This does not apply, however, on the extended sources, as an alternative de Ruiter distance 13 does not suffer from near-360-degree problems.

6.4 Off-database matching

For large number of sources in the field/image matching can be slow in the database. To solve this issue it is possible to activate a F90-based matcher. It is not very efficient, and as such provides no performance increase, at least comparing to PostgreSQL. The improvement is that there is a parallel version of this tiny code making use of the OpenMP technology. It scales almost lineary with the number of CPU cores used (set from console by `export OMP_NUM_THREADS="4"`).

To activate F90-matcher set `matcher=F90` in the settings file (see 6.2), otherwise set it to `SQL`.

6.5 Problem resolution

Column `kind` of `temp_associations` table is then set to the following values:

1 — one-to-one match;

2 — one-to-many match (one observation match many sources in the catalog);

3 — many-to-one match (many observations match to one source in the catalog);

4 — many-to-many match. In this case `group_head_id` column is important. It refers to the smallest ID of the source in the group;

5 — extended sources join proposed, see below;

		Catalog entries		
		0	1	2+
Image sources	0	x	No detection	-
	1	New source	Perfect match (kind=1)	Merged source (kind in (2,5))
	2	New sources	Resolved source (kind=3)	Group (kind=4)

New source/New sources — add sources to database;

Perfect match — add to fluxes and associations tables (update positional information);

Update of the point source (A) by a new observation (X) in a known band (1):

$$A1 + X1 = (AX)1 \tag{14}$$

Update of the point source (A) by a new observation (X) in a new band (2):

$$A1 + X2 = (AX)1, 2 \tag{15}$$

Update of the extended source (K) by a new observation (X) in a known band (1), per-band source (k) is updated:

$$Kk1, l2 + X1 = (KX)(kX)1, l2 \tag{16}$$

Update of the extended source (K) by a new observation (X) in a new band (3), per-band source (m) is created:

$$Kk1, l2 + X3 = (KX)k1, l2, m3 \tag{17}$$

Merged source — add association to the nearest; For point sources (X is closer to A):

$$A1, 2, 3 + B1, 2, 3 + X1 = (AX)1, 2, 3 + B1, 2, 3 \tag{18}$$

For extended sources (X is closer to K):

$$Kk1, l2 + Lk1, l2 + X1 = (KX)(kX)1, l2 + Lk1, l2 \tag{19}$$

Same, but with new band:

$$Kk1, l2 + Lk1, l2 + X3 = (KX)k1, l2, m3 + Lk1, l2 \tag{20}$$

If band sets for matched catalog sources (K and L) are non-intersecting and the observation band is also not in neither of this sets than the two extended sources are merged:

$$Kk1, l2 + Lm3, n4 + X5 = (KLX)k1, l2, m3, n4, o5 \tag{21}$$

Resolved source — see below in section 6.5.1;

$$A1, 2, 3 + X1 + Y1 = (AX)1, 2, 3 + (AY)1, 2, 3 \tag{22}$$

$$Kk1, l2 + X1 + Y1 = (KX)(kX)1, l2 + (KY)(kY)1, l2 \tag{23}$$

No detection — so far ignored;

Group — mark and leave for manual resolution, also see 6.6:

$$A1 + B1 + X1 + Y1 = \text{Group1} \tag{24}$$

6.5.1 Treating resolved sources

For **point sources** a "flux splitting" technique is applied. For example there are two observations in band j with fluxes $f_1^j = 0.2$ and $f_2^j = 0.8$ observed in place where one source with flux $f_0^j = 1$ was observed before. In this case we create a new source (source "b") and copy all existing positional information from the old one (source "a"). Flux information for each band i (including band j) is also copied, but weighted according to fluxes in the new observations:

$$f_a^i = f_0^i \frac{f_1^j}{f_1^j + f_2^j} \tag{25}$$

$$f_b^i = f_0^i \frac{f_2^j}{f_1^j + f_2^j} \tag{26}$$

Then old and new sources are updated with one of the new observations (see equation 22). Positional information from the old source will be updated with (presumably) much more precise information from the new observations, thus the new information will have much higher weight, and final positions will have good quality. "Flux splitting" have to be used for fluxes, as light-curve might be wanted. In this case copying the flux information will lead to

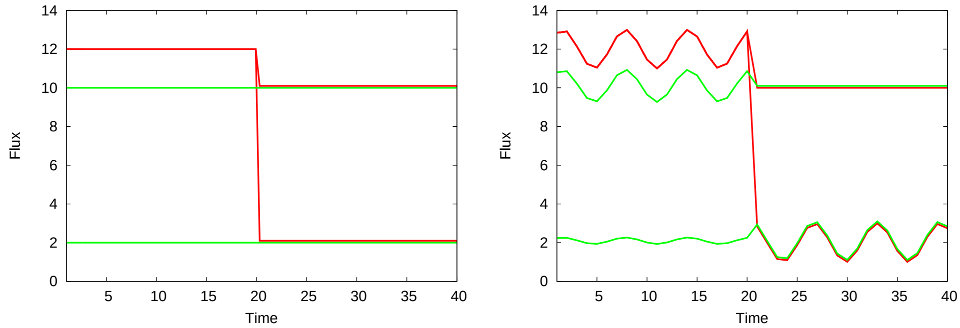

Figure 1: Lightcurves inllustrating "flux splitting" approach. *Left*: two constant sources without (red) and with (green) "flux splitting". *Right*: Same, but fainter source is variable.

instantaneous jumps in fluxes, which might be recognized as ones of transient nature. An example is shown at the figure 1. For constant sources this approach works fine, removing artificial jumps. But if one of the sources is variable, than this variability is split between both sources, which is obviously wrong. Thus for variable sources some post-processing will be required to remove this artifact.

For **extended sources** flux splitting is applied only at per-band level. Only per-band sources are duplicated, while the cross-band source remains the same (see equation 23). Thus several per-band sources with the same band can appear for one cross-band source. This might cause complications, as two different sources might end up having just one cross-band source. This has to be resolved at post-processing. Note, that in the case of extracted sources it is possible, that sum of values in the `datapoints` columns for per-band sources will not match `datapoints` value in the cross-band source. This is due to the fact that both resulting per-band sources (kX and kY in the equation 23) will inherit datapoints from "parent" per-band source (k). Such duplicated datapoints will not be counted twice for cross-band source.

The example of source splitting is shown below. Table 1 shows `runningcatalog` content before splitting. There are two per-band sources with 2 and 1 datapoints and a corresponding cross-band source with 2+1=3 datapoints. Then there was a detection of two objects near per-band source with id=2. This source was splitted in two, having ids 2 and 4, with 3 datapoints each. Cross-band source was also updated, but only 2 datapoints were added from 2 new detections. Thus the sum of per-band datapoints is greater than the number of cross-band datapoints, see table 2.

runcatid	band	datapoints	parent_id
1	-	3	-
2	1	2	1
3	2	1	1

Table 1: runningcatalog before extended source splitting

runcatid	band	datapoints	parent_id
1	-	4	-
2	1	3	1
3	2	1	1
4	1	3	1

Table 2: runningcatalog after extended source splitting

6.6 Group resolution

There are several cases when many-to-many problem can be easily solved.

Two algorithms are implemented so far. They might be used from within the pipeline only, so groups already stored in the database cannot be treated so far.

Although, they have to be used with care, as they might produce wrong matching, which might be worth than no match at all.

Two-by-two problem with fluxes If there are two detections matched to two sources with flux information in this band, than we can take this flux information into account. We construct modified de-Ruiter distance:

$$\hat{R}^2_{deRuiter} = R^2_{deRuiter} + \frac{(f_1 - f_2)^2}{(\Delta f_1)^2 + (\Delta f_2)^2}, \tag{27}$$

where $f_{1,2}$ are fluxes and $\Delta f_{1,2}$ are flux errors.

This produces a matrix of distances for sources A, B and detections X, Y:

$$\begin{matrix} \hat{R}_{deRuiter,AX} & \hat{R}_{deRuiter,AY} \\ \hat{R}_{deRuiter,BX} & \hat{R}_{deRuiter,BY} \end{matrix} \tag{28}$$

Here we assign X and Y so that $\hat{R}_{deRuiter,AX} < \hat{R}_{deRuiter,AY}$. We choose as

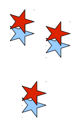

Figure 2: Example of nearest-neighbour group resolution

resolution condition the following:

$$\begin{cases} \hat{R}_{deRuiter,AX} < \hat{R}_{min} \\ \hat{R}_{deRuiter,BY} < \hat{R}_{min} \\ \hat{R}_{deRuiter,AY}/\hat{R}_{deRuiter,AX} > P \\ \hat{R}_{deRuiter,BX}/\hat{R}_{deRuiter,BY} > P \end{cases} \tag{29}$$

This means that in coordinate-flux space detection X is P times closer to A then to B, and detection Y is P times closer to B then to A. Thus we state that A matches X and B matches Y. For the current implementation $P = 10$.

Nearest neighbour In this approach we can treat N-to-N problems, i.e. when the number of confused detections and sources is equal. We find for each detection and source it's nearest neighbour amongst sources and detections respectively. The resolution criteria is then:

- each detection has a mutual nearest neighbour source at de Ruiter distance d and
- there is no other source within $50d$.

An example of such situation (distances between pairs are not to scale) is shown in figure 2.

6.7 Spectral information

Stored in the database as polinomia coefficients calculated according to [4]. Last date of spectral fitting is stored as well. Fits are done upon request, if there are detections newer then the last fit performed.

6.8 Reprocessing and image status

Reprocessing is possible. There are two columns to indicate reprocessing in the `images` table. `status` shows the status of this image. On creation (before any processing was done), status is 0. After the processing stage is completed, status is changed to 1. If reprocessing is done, than all data coming from a given image is removed first from `runningcatalog`, `runningcatalog_fluxes` and `assocxtrsources` tables and status is set to 2. If needed, original data from `extractedsources` is also removed and image status is then changed to 3.

After removal a second stage of reprocessing is data processing itself. It is done in the same way as the first-time processing. When this stage is completed, status is set to 1 and counter in the `reprocessing` column is incremented.

7 Development and Deployment

7.1 Coding rules

For Python use PEP8 (http://www.python.org/dev/peps/pep-0008/), wherever possible. For SQL — to be defined basing on the existing code.

7.2 Database structure

frequencybands		
freqbandid	integer	-
freq_central	float	Band middle frequency (Hz)
freq_low	float	Band lower limit frequency (Hz)
freq_high	float	Band upper limit frequency (Hz)

datasets		
dsid	integer	-
rerun	integer	-
dstype	integer	-
process_ts	timestamp	-
dsinname	character(64)	-
dsoutname	character(64)	-
description	character(100)	-

runs		
runid	integer	-
start_date	timestamp	Time of run begin
end_date	timestamp	Time of run end
status	integer	Run status (...)
user_id	character(100)	User who started a run
process_id	integer	System process id

images		
imageid	integer	-
ds_id	integer	-
tau	integer	-
band	integer	Frequency band
stokes	character(1)	One of IQUV
imagename	character(64)	LOFAR image ID
centr_ra	float	Pointing ra (degrees)
centr_decl	float	Pointing decl (degrees)
fov_radius	float	Field of view (degrees)
bmaj	float	Beam major axis (degrees)
bmin	float	Beam minor axis (degrees)
bpa	float	Beam pitch angle (degrees)
url	character(120)	Image file name
reprocessing	integer	Number of reprocessings of this image
status	integer	Image status (...)
process_date	timestamp	Last image processing date
svn_version	integer	SVN version of the pipeline of the last processing
run_id	integer	-

extractedsources		
xtrsrcid	integer	-
xtrsrcid2	integer	Reference to xtrsrcid for sources mirrored across 360-degrees
image_id	integer	-
zone	integer	integer part of decl
healpix_zone	integer	Zone of HEALpix division
ra	float	right ascention (degrees)
decl	float	declination (degrees)
ra_err	float	error of ra
decl_err	float	error of decl
x	float	x-coordinate on 3D unit sphere
y	float	y-coordinate on 3D unit sphere
z	float	z-coordinate on 3D unit sphere
det_sigma	float	detection threshold
source_kind	smallint	0=point source, 1=extended source
g_major	float	major axis for extended source
g_major_err	float	error of major axis for extended source
g_minor	float	minor axis for extended source
g_minor_err	float	error of minor axis for extended source
g_pa	float	positional angle for extended source
g_pa_err	float	error of positional angle for extended source
f_peak	float	Peak flux
f_peak_err	float	Peak flux error
f_int	float	Integrated flux
f_int_err	float	Integrated flux error

assocxtrsources		
xtrsrc_id	integer	Reference to extractedsources
runcat_id	integer	Reference to runningcatalog
weight	float	Association weight
distance_arcsec	float	Distance in arcseconds
lr_method	integer	-
r	float	de Ruiter distance
lr	float	-

detections		
run_id	integer	-
image_name	character(64)	-
lra	float	-
ldecl	float	-
lra_err	float	-
ldecl_err	float	-
lf_peak	float	-
lf_peak_err	float	-
lf_int	float	-
lf_int_err	float	-
g_minor	float	-
g_minor_err	float	-
g_major	float	-
g_major_err	float	-
g_pa	float	-
g_pa_err	float	-
ldet_sigma	float	-
healpix_zone	integer	-

runningcatalog		
runcatid	integer	-
first_xtrsrc_id	integer	Id of the first observation
ds_id	integer	-
band	integer	Frequency band for per-band extended source
stokes	character(1)	Stokes parameter for per-band extended source
datapoints	integer	Number of observations
decl_zone	integer	integer part of decl
healpix_zone	integer	Zone of HEALpix division
wm_ra	float	-
wm_ra_err	float	-
avg_wra	float	-
avg_weight_ra	float	-
wm_decl	float	-
wm_decl_err	float	-
avg_wdecl	float	-
avg_weight_decl	float	-
source_kind	smallint	0=point source, 1=extended source
parent_runcat_id	integer	Id of the cross-band source for per-band extendedsource
wm_g_minor	float	-
wm_g_minor_err	float	-
avg_wg_minor	float	-
avg_weight_g_minor	float	-
wm_g_major	float	-
wm_g_major_err	float	-
avg_wg_major	float	-
avg_weight_g_major	float	-
wm_g_pa	float	-
wm_g_pa_err	float	-
avg_wg_pa	float	-
avg_weight_g_pa	float	-
is_group	boolean	True if a source belongs to a group
group_head_id	integer	Group ID
deleted	boolean	True if this source was deleted
last_update_date	timestamp	Date of the last source update
x	float	x-coordinate on 3D unit sphere
y	float	y-coordinate on 3D unit sphere

runningcatalog_fluxes		
runcat_id	integer	Reference to runningcatalog
band	integer	Frequency band
stokes	character(1)	Stokes parameter
datapoints	integer	Number of observations
wm_f_peak	float	Peak flux
wm_f_peak_err	float	-
avg_wf_peak	float	-
avg_weight_f_peak	float	-
wm_f_int	float	Integrated flux
wm_f_int_err	float	-
avg_wf_int	float	-
avg_weight_f_int	float	-

temp_associations		
xtrsrc_id	integer	-
xtrsrc_id2	integer	-
runcat_id	integer	-
distance_arcsec	float	-
lr_method	integer	-
r	float	-
lr	float	-
xtr_count	integer	-
run_count	integer	-
kind	integer	-
group_head_id	integer	-
flux_fraction	float	-
image_id	integer	-

image_stats		
image_id	integer	-
run_id	integer	-
kind	integer	-
lr_method	integer	-
value	integer	-

In temp_associations	
1	Point-point association
2	Extended sources – per-band association
3	Extended sources – cross-band association
In assocxtrsources	
1	Point-point association
2	Extended sources – per-band association
3	Extended sources – cross-band association
4	Copied during source-splitting
5	Group association

Table 3: Values of `lr_method`

7.3 Tests

Unit tests — test that each function is working as desired;

Consistency tests — test that the pipeline and all of it's parts are working;

Stress tests — (or performance tests) test pipeline performance;

7.4 NVSS data load test

As a test, NVSS data was loaded into databases.

	PostgreSQL	MonetDB
Total load time with F90 matcher (seconds)	33	188
Total load time with SQL matcher (seconds)	207	611
Database size after load	93Mb	108Mb

7.5 Deployment and usage

7.5.1 How to install

The next programs/modules are needed to run the GSM pipeline:

MonetDB or PostgreSQL ;

Python tested on version 2.7, but should work on earlier versions as well;

psycopg2 if PostgreSQL is used;

numpy ;

configobj or lofar.parameterset ;

7.5.2 User scripts

cleanup.py Tool to clean all data from the database.

```
Optional arguments:
  -h, --help    show this help message and exit
  -D DATABASE, --database DATABASE
                name of the database to clean
                (default: test)
  -M, --monetdb use MonetDB instead of PostgreSQL
```

recreate_tables.py Drops and recreates all database objects (tables/procedures/views/indices). Database should already exist. Also re-fills `frequencies` table. Usage: same as for `cleanup.py`.

validate_install.py Checks that all required modules are installed.

gsm_pipeline.py Tool to run GSM pipeline for a given parset. Multiple parsets can be listed.

```
usage: gsm_pipeline.py [-h] [-D DATABASE] [-M] [-p] [-q]
                       [filename [filename ...]]

positional arguments:
  filename              list of parset file names

optional arguments:
  -h, --help            show this help message and exit
  -D DATABASE, --database DATABASE
                        database name to load data into
  -M, --monetdb         use MonetDB instead of PostgreSQL
  -p, --profile         add SQL timing output to log
  -q, --quiet           switch console logging off
```

7.5.3 Parallel runs

Running several instances of the GSM pipeline at the same time is possible. Different instances will use the same tables, but will have different `image_id`.

Note, that parallel runs are obviously possible on the non-overlapping fields.

References

[1] HR De Ruiter, HC Arp, and AG Willis. A Westerbork 1415 MHz survey of background radio sources. II-Optical identifications with deep IIIA-J plates. *Astronomy and Astrophysics Supplement Series*, 28:211–293, 1977.

[2] George Heald, A G De Bruyn, Ronald Nijboer, M Wise, and Roberto F Pizzo. The LOFAR Multifrequency Snapshot Sky Survey (MSSS). (October), 2011.

[3] G.H. Heald, G. de Bruyn, R. Nijboer, M. Wise, R. Pizzo, and L. Collaboration. The LOFAR Multifrequency Snapshot Sky Survey (MSSS): Description and First Results. In *American Astronomical Society Meeting Abstracts*, volume 219, 2012.

[4] Anna M. M. Scaife and George H. Heald. A broad-band flux scale for low-frequency radio telescopes. *Monthly Notices of the Royal Astronomical Society: Letters*, pages no–no, March 2012.

[5] Bart Scheers. Creating the initial Global Sky Model. pages 1–2, 2012.

[6] Arno Schoenmakers. MSSS specifications and software requirements. pages 1–7, 2010.