SPECFIND – a tool to search for and identify radio spectra using VizieR radio catalogues

Bernd Vollmer
CDS, observatoire astronomique de Strasbourg

September 3, 2004

1 Introduction

The number of records in radio source catalogues has increased dramatically in the last years. Three major increases are due to R. Dixon's 'Master Source List' in 1981, to the 87GB/GB6/PMN surveys in 1991, and to the release of NVSS, FIRST and WENSS in 1996/1997 (Andernach 1999). On the other hand, the size distribution of electronic radio source catalogues follows a power law near -0.6, i.e. there is a lot of catalogues with a small number of records.

There are 538 catalogues listed in VizieR when choosing the "radio" option. 380 of them contain relevant data for radio cross-identifications. For the moment only a search by parameters of all or a subsample of catalogues is possible. Links between the catalogues are missing. Thus there is a clear need for establishing links between the catalogues and for including the sources into Simbad.

1.1 Radio surveys

There are two main kinds of radio surveys:

- single dish and
- interferometric radio surveys.

Interferometers cannot detect structures in extended sources larger than the size corresponding to the shortest baseline spacing (ten/tens of arcmin for compact configurations). Sources of large extent have smaller structures in them, which will be resolved and detected by interferometers. This makes the identification and unique spectral index determination complex for very extended sources, if observed with single-dish and interferometer telescopes.

The characteristic of radio surveys that have to be taken into account are

- frequency,
- sky coverage,
- angular resolution (beamsize),
- sensitivity.

In the frequency ranges, one can distinguish m/cm radio surveys and mm radio surveys. Systematic surveys are only available in the m/cm wavelength range.

1.2 Radio catalogues

There are in principle two different kinds of radio source catalogues:

- catalogues of all detected radio sources, i.e. systematic surveys and
- catalogues of physical subclasses of the radio sources (e.g. SN remnants, HII regions, etc.). I will call them *sample catalogues*. Out of the 538 catalogues listed in VizieR (when choosing "radio"), about 200 are sample catalogues.

While the first category is more important for VizieR, the second one represents the input for Simbad. In general sources are fitted by 2D Gaussian distributions, which have in principle 4 parameters: (i) the center position, (ii) the major axis, (iii) the minor axis, and (iv) the position angle of the beam. If the source is not resolved, i.e. if it is a point source, the fit corresponds to the beam.

1.3 Cross-identification of radio sources

There are again three categories of cross-identification between radio sources:

- cross-identification based on the proximity between two sources,
- cross-identification based on the radio spectrum,
- cross-identification based on physical characteristics (e.g. galaxies, SN remnants, quasars, AGNs etc.).

The last two kinds of cross-identification are called 'value-added' cross-identification. The present work deals with these cross-identifications using a proximity criterion and a power law radio spectrum. The cross-identification tool is named SPECFIND.

This article is structured in the following way: Sect. 2 describes the normalization of VizieR catalogue tables using mysql. This is necessary, because the VizieR catalogue tables have very different entries. SPECFIND uses its own table standard, which is also described in Sect. 2. The general code structure is discussed in Sect. 3, the code standards are presented in Sect. 4, followed by the detailed code structure (Sect. 5). I have made tests using artificial catalogues, which are discussed in Sect. 6. The code performance, when real radio catalogues are used, is presented in Sect. 8, the results are shown in Sect. 7, and the conclusions are given in Sect. 9.

2 Preparation and uniformization of VizieR radio catalogues

The main difficulty in treating simultaneously different VizieR radio catalogue tables lies in the variety of their table structure. All tables have entries for the names, coordinates, and the flux density of the sources. The name and coordinates can appear in different epochs (B1950 or J2000). Other possible entries are:

- position errors in right ascension and declination (this includes pointing errors and the uncertainty with which the center of the 2D Gaussian could be determined),
- error of flux density,
- major/minor axis of the source of the fitted ellipse,
- position angle of the fitted ellipse,
- different sorts of flags (warning, border of the observed field, possible confusion, etc.).

This variety of entries makes a uniformization of the VizieR radio catalogues unavoidable.

SPECFIND uses its own standard for the VizieR catalogue entries. All coordinates are in J2000. The source names are chosen to be in accordance with the Simbad nomenclature. Tab. 1 shows the list of SPECFIND catalogue entries. The integrated flux densities are taken directly from the radio catalogues,

Table 1: SPECFIND catalogue standard.

RA(2000)	arcsec		REAL
RA_error	arcsec		REAL
DEC(2000)	arcsec		REAL
DEC_error	arcsec		REAL
Flux	mJy		REAL
flux_error	mJy		REAL
Wflag	warning	0/1	TINYINT
Cflag	confused source	0/1	TINYINT
Eflag	extended source	0/1	TINYINT
MajAxis	major axis (units of beamsize)		REAL
MinAxis	minor axis (units of beamsize)		REAL
PA	position angle (degrees)		REAL
name2000	source name J2000		VARCHAR(20)
name1950	source name B1950		VARCHAR(20)

except for the GB6, 87GB, and MIYUN catalogues. In the GB6 and 87GB catalogues only peak flux densities are given. We take these values if major and minor axis are smaller than 1.1 times the beamsize ($3.5'$). If this is not the case the integrated flux density is obtained by Flux$\times$MajAxis$\times$MinAxis. For the MIYUN catalogue we take the peak flux density as integrated flux density whereever the latter is not given explicitly. The values of the flux errors that are not directly taken from the catalogues are listed in table 2.

The flags are based on those taken from the catalogues. For the moment SPECFIND does not make use of the different flags.

The procedure of the catalogue normalization is done using a **linux mysql** server:

- chose a catalogue in VizieR (webpage),
- take columns that correspond to SPECFIND standard columns (flags might have to be transformed),
- ask for the tabulated output with ';' separations. Rename the file and save it in the appropriate directory,
- cut and copy the file header into a new file called 'header.txt',
- do not forget to delete the last lines of the VizieR output catalogue,
- look for nomenclatura of the sources on the CDS webpage,
- create a raw table within mysql according to the VizieR table and to the table standard (Tab. 1),
- remember: flux density in mJy, RA_error and DEC_error in arcsec!!!
- create standard table according to the table standard (Tab. 1); the source right ascensions and declinations are converted into arcsec; negative declinations beginning with a zero are taken care of,

Table 2: Definitions of the flux error when not taken directly from the catalogues.

JVAS	$0.1 \times$ Flux
BWE	$\sqrt{36. + (0.01 \times \mathrm{Flux}^2)}$
PKS	Flux/SN
PKS	$0.2 \times$ Flux
F3R	error code
B2	$0.2 \times$ Flux
B3	$0.2 \times$ Flux
TXS	$0.2 \times$ Flux
WISH	flux_error or $0.15 \times$ Flux
WENSS	flux_error or $0.15 \times$ Flux
MIYUN	peak flux or Flux/SN
4C	$0.3 \times$ Flux
4C	$0.3 \times$ Flux

- look for and correct for doubles, triple, etc. names,
- run test.sql: 'mysql < test.sql > out.dat',
- replace first line by: %i (number of sources) %f (frequency in MHz) %f (resolution in arcmin).

The final output is an ASCII file that can be read directly by SPECFIND.

3 SPECFIND: general code structure

The code is written in **C**. The search for neighbouring sources is done using 2D treecode routines written by J. Barnes, University of Hawaii (see Barnes & Hut 1986). This method decreases the number of calculations to find the next neighbours of a source from N^2 to $N \log N$. The routines are:

- code.h, defs.h, real.h, stdinc.h, vectmath.h: definitions,
- util.c: utilities used by the treecode,
- load.c: builds up the tree,
- grav.c: proximity search written by J. Barnes, Hawaii, adapted by B. Vollmer (see below).

In addition, SPECFIND uses the method of the least absolute deviation to make a linear fit in the $\log \nu$–$\log S_\nu$ plane, where ν is the frequency and S_ν is the flux density at the frequency ν. This method is more robust against outlyers than a standard least square deviation (χ^2) fit. The routines are taken from 'Numerical Recipes in C' (Press et al. 2002):

- nrutil.h: definitions used in 'Numerical Recipes',

- util.c: routines and definitons of 'Numerical Recipes' used in medfit.c,
- medfit.c: method of minimizing the least absolute deviations from 'Numerical Recipes' chapter 15.7.

SPECFIND is a hierachical code. It classifies a source j as parent, sibling, or child with respect to a given source i at different stages:

1. stage: depending on proximity criteria:
   - parent: source j has a larger extent or resolution than source i,
   - sibling: source j has a comparable extent or resolution (within 25%) to that of source i,
   - child: source j has a smaller extent or resolution than source i.
2. stage: depending on the flux densities at the same frequency:
   - parent: source j has a larger extent or resolution and has a larger flux density than source i,
   - sibling: source j has a comparable extent or resolution and has the same flux density within the errors as source i,
   - child: source j has a smaller extent or resolution and a smaller flux density than source i.
3. stage: depending on flux densities at different frequencies based on the radio spectrum:
   - parent: source j has a larger flux density than expected from the radio spectrum that includes source i,
   - sibling: source j fits into the radio spectrum that includes source i,
   - child: source j has a smaller flux density than expected from the radio spectrum that includes source i.

At the end of this procedure source i and its siblings are the same source. If there are parents, source i might be a resolved subsource of source j. If there are children, source i might be unresolved and the children represent resolved sub-sources of source i. SPECFIND also adds the flux densities of the children of source i at the same frequency. If the sum equals the flux density of source i within the errors, source i is considered as an unresolved source.

The main program (main() in maingal.c) allocates space for the sources and calls the different tasks to treat the data:

- config() in configold.c: reads the catalogue files,
- maketree(bodyptr, int) in load.c: builds up the tree,
- hackgrav(bodyptr, bool) in grav.c: proximity search,
- joinsource(bodyptr) in joinsources.c: takes into account the source flux densities at the same frequency,
- crosssource(bodyptr) in crosssources.c: checks for family dependence (see below),

- compresssource(bodyptr) in compresssources.c: removes sources which are parents, siblings, children of parents, siblings, children.
- spectra(bodyptr) in spectra.c: restores parent, sibling, child dependences (decompressing), extracts spectra using the method of minimizing the least absolute deviation,
- speccheck() in speccheck.c: security task: checks if source i fits in found spectrum; checks for frequency spacing,
- spreadspec() in spreadspec.c: completes spectra, i.e. if source j appears in the radio spectrum of source i, source i must also appear in the radio spectrum of source j,
- output() in configold.c: output into ASCII file.

4 Code standards

As described above SPECFIND uses its own standard for the input files. Since the VizieR catalogues are by far not uniform, it was unevitable to define a standard where all parameters necessary for the spectrum finding algorithm are represented. Each catalogue has its input ASCII file with the following structure:

header:
%i %f %f total number of sources, frequency (MHz), resolution (arcmin);
entry for each source:
%f %f %f %f %f %f %d %d %d %f %f %f %s
RA (sec),
RAerror (arcsec),
DEC (arcsec),
DECerror (arcsec),
Flux (mJy),
Fluxerror (mJy),
Wflag: warning (0/1),
Cflag: confusion (0/1),
Eflag: extension (0/1),
Majaxis (in units of resolution),
Minaxis (in units of resolution),
PA (degrees),
Name.

For the moment SPECFIND does not use the pointing errors (RAerror, DECerror) and flags (Wflag, Cflag, Eflag). Note: within SPECFIND the RA has units of arcseconds. Since the position angle is defined as the angle rotating eastwards from the north, one has to use $90^{o} - PA$ for the position angle in SPECFIND.

The number of catalogues and the catalogue specifications are contained in the ASCII file 'catalogues.dat'. Its structure is the following:

header:

%i number of catalogues
entry for each catalogue:
%s %f %f %f %f
catalogue name,
frequency (MHz),
resolution (arcmin),
largest source extent (arcmin),
total number of sources.

If one wants to add a new catalogue, it has to be prepared following the procedure described in Sect. 2, placed in the correct directory, and added as a new line in 'catalogue.dat'.

Inside the code, each source is represented by a structure where all necessary information is stored (defs.h):

typedef struct :
vector pos → coordinates (J2000);
real majaxis → major axis;
real minaxis → minor axis;
real pa → position angle;
real frequency; → frequency
real resolution; → resolution
real raerror; → pointing error
real decerror; → pointing error
real flux; → flux density
real fluxerror; → flux density error
real specindx; → spectral index
real absz; → spetrum absizza
short resolved; → frequency where source is resolved
short wflag; → warning
short cflag; → source confusion
short eflag; → extended source
char name[22]; → source name
short nparents; → number of parents
short nbrothers; → number of siblings
short nsons; → number of children
nodeptr parents[50]; → pointers (adresses) of parents
nodeptr brothers[50]; → pointers (adresses) of siblings
nodeptr sons[50]; → pointers (adresses) of children
bool done; → flag to identify unique sources
body, *bodyptr;

ATTENTION: the number of parents, siblings, children is limited to 50 (defs.h), the number of sources is limited to 500,000, the number of sources per spectrum is limited to 1000, and the number of catalogues is limited to 1000.

5 Detailed code structure

In this section all routines mentioned in Sect. 3 are described in detail. All specifications are given for a linux PC with 512 MB RAM and a frequency of 1.4 GHz.

5.1 Main routine (main() in maingal.c)

The main routine allocates space for the source structures described in Sect. 4. The maximum value of sources depends on the RAM capacity of the PC. For 512 MB the maximum number of soucrces is 500,000. In a second step the routine reads the catalogue list 'catalogues.dat'. As described in Sect. 4 the number of catalogues and all necessary information about the catalogues are stored in this ASCII file.

Since the number of sources that can be treated at the same time is limited, SPECFIND divides the sky into RA slices. The number of slices is set in a way that the number of sources located in one slice does not exceed 500,000. The number of files is stored in the variable 'filenum'. Let $a_{\rm min}/a_{\rm max}$ be the right ascension of the western/eastern border of the slice using the above mentionned subdivision, then the actual slice borders are $a_{\rm min} - 3\,r$ and $a_{\rm max} + 3\,r$, where r is the maximum extent of all sources in all catalogues and the factor 3 is a security factor. For the final source list, only the sources within $a_{\rm min}$ and $a_{\rm max}$ are taken into account. The subsequent data processing is the same for all slices. The main routine then calls the routine for building the tree ('maketree') and the routine of the proximity search for each particle ('hackgrav(bodyptr, bool)'; Sect. 5.2). A check for sources located around RA = $00^{\rm h}00^{\rm m}00^{\rm s}$ is included. After the proximity search the main routine calls 'joinsource(bodyptr)' (Sect. 5.3), 'crosssource(bodyptr)' (Sect. 5.4), and 'compresssource(bodyptr)' (Sect. 5.5). The heart of the code is the routine 'spectra(bodyptr)', which searches for and identifies radio spectra (Sect. 5.6). After the spectrum finding algorithm a security check is done ('speccheck()'; Sect. 5.7), before the spectra are checked for completeness with respect to the other spectra ('spreadspec()'; Sect. 5.8). This is followed by a second security check ('speccheck()'; Sect. 5.7). At the end of each RA slice processing a list of numbers is printed out:

- 'number of output files': number of RA slices,
- 'RA from %f to %f arcseconds': borders of RA slices overlap included,
- 'reading catalogue %s': name of the catalogue read (for each RA slice all catalogues are read),
- 'begin maketree with %i particles': number of particles within the RA slice overlap included,
- 'time used for search: %i sec': elapsed time for the proximity search,
- '%i parents %i brothers %i sons found': total number of identified parents, siblings, children,
- '%i multiple coordinates found': number of cases where more than two sources with the same coordinates were found (in 'load.c'),

- '%i brother sources joined': number of joined sources (see Sect. 5.3),
- '%i time used for joining: %i sec',
- '%i sources dropped when crosschecked': number of sources that where removed, because the check for the familiy dependence failed (see Sect. 5.4),
- 'time used for crossing: %i sec': elapsed time for family dependence check,
- '%i resolved sources': number of resolved sources at the same frequency,
- '%i sources dropped when compressed': number of parents, siblings, children that were dropped, because they are redundant (see Sect. 5.5),
- '%i double sources found': number of parents, siblings, children that are the same as the source itself,
- 'time used for compressing: %i sec': elapsed time for source compression,
- 'time used for spectra: %i sec': time needed by the spectrum finding algorithm,
- 'number of spectra found: %i': total number of spectra found in the RA slice (multiple sources included),
- 'number of spread spectra: %i': number of spectra that where replaced by a more complete spectrum of a sibling,

After the processing of all RA slices the results are written out as ASCII files (output(); Sect. 5.9) and some important numbers are given:

- 'total number of sources': total number of sources in all catalogues,
- 'processed sources': number of sources that were treated (sources in slice overlaps included),
- 'number of output sources': total number of sources that have at least one parent or sibling or child (at the same or at a different frequency),
- 'total number of spectra found': total number of spectra found (not unique spectra, spectra of sources in slice overlaps included),
- 'final number of spectra found': total number of spectra found (not unique spectra),
- 'total number of associations': number of independent sources that have at least one parent or sibling or child (at the same or at a different frequency),
- 'total number of independent spectra': number of independent sources that are part of a spectrum with more than two independent points (see Sect. 5.6),
- 'total number of independent flat spectra': number of independent sources that have a spectral index α greater than -0.5. The radio spectrum is defined as

  $$S_\nu \propto \nu^\alpha \tag{1}$$

  where ν is the frequency, S_ν is the flux density at the frequency ν, and α is the spectral index.

5.2 Proximity search (maketree(bodyptr, int), hackgrav(bodyptr, bool) in grav.c)

The proximity search is done using the treecode routines written by J. Barnes, University of Hawaii (see e.g. Barnes & Hut 1986; http://www.ifa.hawaii.edu/barnes/software.html). Since this code is well documented, I will not describe the method nor the routines. The only routine that I have adapted is the one defining the acceptance criterion for a neighbouring source ('gravsub(nodeptr)' in 'grav.c').
Let d_1=max(resolution(i),Majaxis(i)) and d_2=max(resolution(j),Majaxis(j)), i.e. the maximum between the resolution, with which the source i/j was observed and their major axis. If $d_2 > 1.25\, d_1$, source j is considered as a parent. If $0.75\, d_1 \leq d_2 \leq 1.25\, d_1$, source j is considered as a sibling. If $d_2 < 0.75\, d_1$, source j is considered as a child. An additional search for sources that are loacted around around RA = 00 00 00 is included.
The angular distance between two sources, which are located at (α_1, δ_1) and (α_2, δ_2) is calculated in spherical geometry:

$$d = \arccos(\cos(\alpha_1 - \alpha_2)\cos(\delta_1)\cos(\delta_2) + \sin(\delta_1)\sin(\delta_2)) \ . \tag{2}$$

Since the treecode works in plane geometry, the polar caps have to be treated separately. The rest of the sky is divided into equidistant RA slice. This ensures an approximately equal number of sources in each slice. For the next neighbour search within each slice, sources within a larger field are used. The RA offset between this field and the slice is taken to be three times the larest source extent within the slice.
The largest extent of all sources is $1^{\rm o}$, and thus the maximum overlap is $3^{\rm o}$. At a declination of 70 degrees a linear separation of $(\alpha_1 - \alpha_2) = 3^{\rm o}$ in right ascension corresponds to an angular separation of $1^{\rm o}$. Thus, the two polar caps with $|\delta| > 70^{\rm o}$ are treated as two separate regions to insure the correct treatement of the spherical geometry by the code. The proximity search within both polar regions were effected without using the treecode ($N \times N$ calculations). There was no change of the result with respect to the treecode.
For the check if source j is located within the ellipsoidal source extent of source i, the ARC projection is used. In this projection angular distances are preserved. It is used in Schmidt telescopes (to first order) and in mapping with single dish radio telescopes. On the other hand, for interferometric maps the SIN projection is applied. Since (i) the majority of our catalogues are single dish measurements, (ii) the largest source extents are found in single dish observations because of their larger beamsize, and (iii) since the ARC and the SIN projection are close to each other, I have adopted the ARC projection.

5.3 Joining of sources at the same frequency (joinsource(bodyptr) in joinsources.c)

This routine takes into account the flux densities of the sources at the same frequency. Since the source extent frequently equals the beamsize/resolution of the observations (the sources are not resolved), a source observed with a given beamsize is classified as a parent of another source observed at the same frequency with a smaller beamsize. Consequently, a source observed with a given

beamsize is classified as a child of another source observed at the same frequency with a larger beamsize. However, it is possible that both sources are the same physical object. This can be verified with the help of the flux densities measured at the same frequency. The routine makes the following classification:
(i) parent: larger flux density and larger extent/resolution,
(ii) sibling: equal flux density within the errors and equal extent/resolution within 25%,
(iii) child: smaller flux density and smaller extent/resolution.

5.4 Check for family dependences (crosssource(bodyptr) in crosssources.c)

Once the sources are joined in the way descibed in Sect. 5.3, the family dependences are verified, i.e. for a given source cross-checks are performed. These checks are performed for all source entries;

- If source j is a sibling of source i, source i must also be a sibling of source j.
- If source j is a child of source i, source i must be a parent of source j.
- If source j is a parent of source i, source i must be a child of source j.

In addition, the flux densities of all children are added to investigate if the source has multiple components at a given frequency. If the sum of the flux densities of all siblings equals that of the source within the errors, it is considered as resolved and the corresponding flag ('Wflag(bodyptr)') is activated. It happens frequently that a non-Gaussian emission distribution consists of multiple components within the area of one beamsize.

5.5 Source compression (compresssource(bodyptr) in compresssources.c)

This routine removes sources, which are parents/siblings/children of parents/siblings/children. Within the spectra finding algorithm (Sect. 5.6) these dependences are restored. This procedure, which might appear laborious, has the advantage that even if the other routines missed a source, it will be most probably recovered by the restoring process.

In detail the routine checks if
(i)
a parent of a sibling is found between the siblings;
a sibling of a sibling is found between the siblings;
a child of a sibling is found between the siblings;
a parent of a sibling is found between the parents;
a sibling of a sister/bother is found between the parents;
a child of sibling is found between the parents;
s parent of s sibling is found between the children;
a sibling of a sibling is found between the children;
a child of a sibling is found between the children;
(ii)

a parent of a parent is found between the siblings;
a sibling of a parent is found between the siblings;
a child of a parent is found between the siblings;
a parent of a parent is found between the parents;
a sibling of a parent is found between the parents;
a child of a parent is found between the parents;
a parent of a parent is found between the children;
a sibling of a parent is found between the children;
a child of a parent is found between the children;
(iii)
a parent of a child is found between the siblings;
a sibling of a child is found between the siblings;
a child of a child is found between the siblings;
a parent of a child is found between the parents;
a sibling of a child is found between the parents;
a child of a child is found between the parents;
a parent of a child is found between the children;
a sibling of a child is found between the children;
a child of a child is found between the children.
In these cases the dependent sources are removed. In addition, a search for double parents, siblings, children is performed. The doubles are also removed.

5.6 Spectra finding algorithm (spectra(bodyptr) in spectra.c)

This is the most important routine and thus the heart of SPECFIND. It extracts spectra using the 'Numerical Recipes' routines in 'medfit.c'. As described in Sect. 3 these routines are the implementation of the method of the least absolute deviation.

For this algorithm the flux density errors of all sources that are smaller than 20% of their flux density are set to this value. Furthermore, the flux density errors are multiplied by a factor 1.5. In this way all catalogues have approximately the same relative error. Moreover, these relatively large errors can compensate a not too strong flattening of the spectral index at low frequencies due to the synchrotron turnover or at high frequencies due to an increasing fraction of emitting thermal electrons. This choice of the flux density error turned out to be the best way to find a maximum number of spectra without a too high risk of misidentifications.

In a first step, the parent/sibling/child dependences are restored (parents of siblings, children of siblings, parents of parents, childs of parents, parents of childs, children of children). Siblings of the same frequency as the source itself are stored separately. A check for sources located around RA = 00 00 00 is performed.

The sources at different frequencies are then stored into an array and sorted by frequency. If the number of sources at different frequencies is greater than two, the spectrum finding algorithm is started:

1. a least absolute deviation fit in the $\log S - \nu - \log \nu$ plane is performed:

$$\log S_\nu = b \log \nu + a \ ; \qquad (3)$$

2. if the number of sources that fit into the spectrum decreases, the old fit parameters are used;

3. check for sources that fit into spectrum; if all sources fit, goto 6.;

4. if there are two sources of the same frequency, the one with the largest deviation from the fit is flagged and removed;

5. if all sources have different frequencies the source with the largest deviation from the fit is flagged and removed, goto 1.;

6. if there are more than two independent points left, make a final fit;

7. goto 1. and make a second run with $b = -0.9$ and $a = \log S_\nu - b \log \nu$ during the first $N - 4$ steps of the loop, where N is the initial number of points in the spectrum (-0.9 is the mean spectral index of all radio sources);

8. if the number of fitted points with fixed a and b exceeds that of the initial fitting prodecure, this spectrum is accepted; otherwise the spectrum of the first fitting procedure is accepted.

This algorithm turned out to be the most promising in order to find a maximum number of spectra with only a small risk of misidentifications (see Sect. 7)

In order to investigate the efficiency of the spectrum finding algorithm, the data of the sources, for which no spectrum could be found are written in an ASCII file named 'Abfall1.dat', whose format is the following:
%s name of the source
%i %f %f number of points of the spectrum, spectrum abszissa a, spectral index b (Eq. 3).
%f %f %f frequencies, flux densities, and flux density errors of all points of the sectrum.
This ASCII file can be directly read by the IDL program 'abfall.pro', which makes plots of these points. In this way it is possible to investigate why the spectrum finding algorithm failed. With the help of this tool and the tool 'specplot.pro' (Sect. 5.9) I optimized the code to find a maximum number of radio spectra with a relatively small number of misidentifications.

It turned out that source confusion is a big problem, especially for sources that are observed with very different resolutions (more than a factor 10). Let $r_{\rm max/min}$ be the largest/smallest source extent/resolution. I decided to reject a spectrum made of 3 points with a spectral index smaller than -2 and $r_{\rm max}/r_{\rm min} > 10$.

5.7 Security check (speccheck() in speccheck.c)

This rather small routine insures that all points fit into the spectrum determined by 'spectra(bodyptr)' (Sect. 5.6), especially that the source itself fits into the spectrum. If there are points that do not fit, they are removed from the spectrum. If at the end there are less than three independent points left, the spectrum is dropped.

In a second step the sources in the spectrum are checked for proximity, i.e. their extents or resolutions have to intersect. If this is not the case, the spectrum is dropped.

In a third step the frequency intervals between the different points of the spectrum are checked. In this way spectra are excluded, which are based on points that are not independent, because some of their frequencies are too close. In practice, the routine calculates the frequency intervals and determines the ratio between the second largest and the largest frequency interval. If this ratio is smaller than 0.02 the spectrum is rejected.

In a fourth step, ambiguous sources are detected. These are radio sources at a given frequency that are attributed to two distinct physical objects. Whenever, such a source is identified, the center of mass coordinates for both objects are calculated, where the mass is substituted by one over the resolution. The ambiguous source is then attributed to the object with the smallest distance between the source position and the center of mass position of the object. The amiguous source is deleted from the second object.

5.8 Completeness and uniqueness check for spectra (spreadspec() in spredspec.c)

This routine insures that if a source j fits the spectrum determined for the source i (where the source i is included), the source i also appears in the spectrum of the source j. In this way it is insured that a radio source belongs to only one single physical object. This routine is equivalent to the routine 'crosssource(bodyptr)' described in Sect. 5.4, but it handles the dependences at different frequencies.

In practice the spectrum finding algorithm is too efficient. Since the flux density errors and beamsizes are quite large, by chance alignments where the frequency differences are small at low frequency and where there is only one point at a high frequency are identified as physically belonging together (via the radio spectrum). Thus it happens that if a source j fits the spectrum determined for the source i, the source i is not included into the spectrum of source j. This can have several reasons: (i) the spectrum of source i is erroneous, (ii) the spectrum of source j is erroneous, (iii) the spectral index varies with frequency, (iv) the real errors on the flux densities of one of the sources is larger than 30%, The task to make all spectra consistent is quite complicated, because all sources are interconnected via their siblings, which are again interconnected via their siblings etc.

In a first step, the number of independent points of the spectra of source i and j are compared. If the number of points of the spectrum of source i exceeds that of the spectrum of source j and source j has no identified spectrum, the spectrum of source j is replaced by the spectrum of source i and vice versa.

In order to decide which spectrum to take if there is inconsistency between two spectra, the following scheme is applied:
Let $\alpha_{i,j}$ be the spectral indices of source i and j. (i) If the difference between the spectral indices is smaller than 0.3 ($|\alpha_i - \alpha_j| < 0.3$) (see Vollmer et al. 2004) both spectra are real.
(ii) If ($|\alpha_i - \alpha_j| \geq 0.3$) and the frequency intervalls within which they are determined are only marginally overlapping (20%) both spectra are real.
(iii) if (i) and (ii) are not the case, the spectrum with the larger number of independent frequency points is real. The other spectrum is discarded.

(iv) if (i) and (ii) are not the case and numbers of independent frequency points are the same for both spectra,
(iva) if $\alpha_i < 0$ and $\alpha_j < 0$ and $\alpha_i > \alpha_j$, spectrum i is real and spectrum j is discarded;
(ivb) if $\alpha_i < 0$ and $\alpha_j < 0$ and $\alpha_i < \alpha_j$, spectrum j is real and spectrum i is discarded;
(ivc) if $\alpha_i \times \alpha_j < 0$ and $\alpha_i < 0$, spectrum i is real and spectrum j is discarded;
(ivd) if $\alpha_i \times \alpha_j < 0$ and $\alpha_j < 0$, spectrum j is real and spectrum i is discarded;
(ive) if $\alpha_i > 0$ and $\alpha_j > 0$ and $\alpha_i < \alpha_j$, spectrum i is real and spectrum j is discarded;
(ivf) if $\alpha_i > 0$ and $\alpha_j > 0$ and $\alpha_i > \alpha_j$, spectrum j is real and spectrum i is discarded.

The sources in the spectrum are checked for proximity, i.e. there extents or resolutions have to intersect. It turned out that source confusion is a big problem, especially for sources that are observed with very different resolutions (more than a factor 10). Let $r_{\rm max/min}$ be the largest/smallest source extent/resolution. I decided to reject a spectrum made of 3 points with a spectral index smaller than -2 and $r_{\rm max}/r_{\rm min} > 10$.

The above described procedure checks only for siblings of siblings. This does not take into account the case where the same source is attached as a sibling to two different sources without being a sibling of a sibling of any of these two sources. The only way to identify these cases is to make again a full proximity search using the treecode (Sect. 5.2). All near sources are then checked common siblings. If the modulus of the difference between the spectral indices of the two sources is smaller than 0.3, the missing siblings are added to both sources if the proximity criterion is met (the distance is smaller than half the sum of the source extents/resolutions). If the modulus of the difference between the spectral indices of the two sources is greater or equal to 0.3, the siblings of sources i and j are modified in the following way: let the number of siblings of source i be greater than that of source j. If the numbers of siblings are the same, the source with the steeper spectrum is source i. If source j is a sibling of source i, this sibling is removed. In addition, all siblings are removed from source j that have a different frequency than source j. If source j is not a sibling of source i, the common sibling is removed from source j.

In a final step the siblings of a given source are compared to the siblings of all its siblings. If there are less than three common siblings and both spectral indices are non zero and the numbers of siblings are different, the spectrum of the source with the smallest number of siblings is removed (i.e. the siblings at a different frequency than the source are removed). If this is not the case, both sources are complemented with the missing siblings.

5.9 Output (output() in configold.c)

At the end of the data processing for one subfield, the results are written out into an ASCII file. Since all necessary informations for all sources are stored in the structures described in Sect. 4, the output format can be chosen freely. I have adopted the following three output formats:

1. Integral output:

data of all sources that have at least one parent or sibling or child are written into files named 'Datei#.dat', where # is the number of the RA slice. The output format is:
%s   %s %s %s ...
name of the source, names of the parents,
%f %i   %s %s %s ...
spectral index, frequency at which the source might be resolved, name of the siblings,
%s %s %s ...
names of the childs.
I would like to emphasize that at this point of the data processing the siblings j are the same physical object as the source i.

2. Integral or unique output for spectra plotting program:
all data and data of all unique sources that have at least one parent or sibling or child are written into ASCII files. The output file of unique sources is named 'Spektren1.dat', that of the integral data 'Spektren2.dat'. In the case of the unique output, one physical object has one entry in this ASCII table. The format of both files is the following:
%s name of the source
%i %f %f number of points of the spectrum, spectrum abszissa a, spectral index b (Eq. 3).
%f %f %f frequencies, flux densities, and flux density errors of all points of the sectrum.

3. Simbad output:
The radio cross-identifications (all siblings) are written in ASCII files named 'Simbad#.dat', where # is the number of the RA slice. These files have the following format:
&I.0 %s name of the source,
%I %s name of names of the siblings,
%J %f %f source coordinates in degrees,
%J.E %f %f 0 pointing errors.
These ASCII files can be directly read by the CDS tool 'raccord'.

The ASCII files 'Spektren1.dat' and 'Spektren2.dat' can directly be read by the IDL program 'specplot.pro', which calculates statistics of the data and makes plots of the spectra (see Sect. 7).

6 Tests

In order to test the SPECFIND, I have generated 5 artificial source catalogues at 4 different frequencies: 4.8 GHz (6 cm), 2.7 GHz (11 cm), 1.4 GHz (20 cm), 600 MHz (50 cm). The second catalogue at 4.8 GHz has a higher resolution than the first one. Ten thousand sources are randomly distributed over the whole sky in each catalogue. At the end the environment for one source was set by hand. SPECFIND had to find the neighbouring sources, to identify correctly the parents, siblings, and children, and to determine the correct spectrum. The extent of the source was enlarged artificially in order to treat an extreme case. Fig. 1 shows the environment of this 4.8 GHz source of low resolution, which is located at the center of the plot. The ellipse delineates the spatial extent of the source. Crosses represent 4.8 GHz sources observed at a low resolution, boxes 4.8 GHz sources observed at a high resolution, stars 2.7 GHz sources, diamonds 1.4 GHz sources, and triangles 600 MHz sources.

Within the source extent there are six additional 4.8 GHz sources of low resolution, two 4.8 GHz sources of high resolution, three 2.7 GHz sources, one 1.4 GHz source, and two 600 MHz sources. The flux densities of the high resolution 4.8 GHz sources were chosen such that their sum equaled the flux density of the central source. SPECFIND was able to find all parents, siblings, and children. It also detected that the source was resolved at 4.8 GHz. In a next step the source was placed near RA = $00^{\rm h}00^{\rm m}00^{\rm s}$, on both sides, in order to check that the periodic boundary conditions are well implemented in the code. SPECFIND handled both configurations correctly.

7 Results

The basis for the list of surveys or systematic observations catalogues can be found in Tab. 1 of Andernach (1999). For the moment I have included the 22 largest of the 66 cited radio catalogues. These catalogues are listed in Tab. 3. The 3C and 3CR catalogues are added for historical reasons. The total number of sources is 3 488 352. Half of them are NVSS sources. For this number of sources the sky is devided into 12 RA slices and 2 polar caps $|\delta| > 70^{\rm o}$.

For the results and the code validation see Vollmer et al. (2004).

8 Code performance

On a PC with 512 MB RAM and a frequency of 1.4 GHz, the whole data processing of 3.5 million sources including input and output is performed in less than 3 h. Less than one quarter of the time is needed to read the input files and to write out the final ASCII files. The proximity search in one single slice is performed within 5-10 min. This relativly long time is due to the spherical geometry (see Sect. 5.2).

9 Conclusions

SPECFIND is a very efficient tool to identify radio spectra using radio catalogues of different formats. SPECFIND can handle radio surveys of very different resolutions and sensitivities. It has been applied to 22 survey csatalogues at 11 different frequencies containing 3.5 million sources, leading to more than 700 000 independent radio cross-identifications and $\sim$ 67 000 indpendent radio spectra with more than two independent frequencies. The code is tested and its results are validated by the comparison between the spectral indices found by SPECFIND and those determined by other authors. The determined spectral indices have an error of about ± 0.3. Negative spectral indices have smaller error while the error of positive spectral indices can be larger, mainly due to a peak of the spectrum within the frequency interval within which the spectrum has been determined. The code is quite rapid (less than 3 h for 3.5 million sources) and since it is written in C, it can be run on virtually all PCs with at least

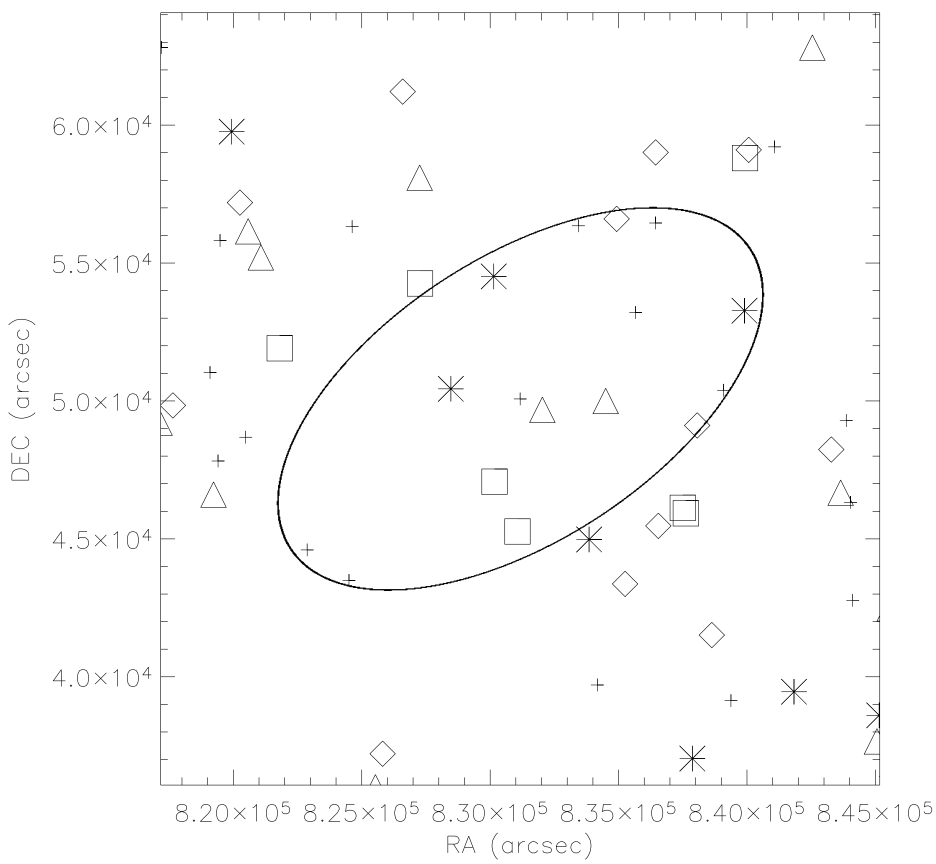

Figure 1: Environment of a low resolution 4.8 GHz source, which is located in the center of the plot. The ellipse delineates the spatial extent of the source. Crosses represent 4.8 GHz sources observed at a low resolution, boxes 4.8 GHz sources observed at a high resolution, stars 2.7 GHz sources, diamonds 1.4 GHz sources, and triangles 600 MHz sources.

Table 3: SPECFIND catalogue entries.

name	Interferometer (I) Single Dish (S)	frequency (MHz)	resolution (arcmin)	S_{min} (mJy)	number of sources	percentage of sources with identified spectrum	Reference
JVAS	I	8400	$5.5\,10^{-3}$	30	2246	72	(1)
GB6	S	4850	3.5	18	75162	59	(2)
87GB	S	4850	3.5	25	54579	64	(3)
BWE	S	4850	3.5	25	53522	61	(4)
PMN	S	4850	3.5	20	50814	30	(5)
MITG	S	4850	2.8	40	24180	52	(6)
PKS	S	2700	8.0	50	8264	65	(7)
F3R	S	2700	4.3	40	6495	61	(8)
FIRST	I	1400	0.08333	1	811117	1.5	(9)
NVSS	I	1400	0.75	2	1773484	3.6	(10)
WB	S	1400	10.	100	31524	60	(11)
SUMSS	I	843	0.75	8	134870	1.8	(12)
B2	I	408	8.0	250	9929	72	(13)
B3	I	408	5.0	100	13340	66	(14)
MRC	I	408	3.0	700	12141	73	(15)
TXS	I	365	0.1	250	66841	57	(16)
WISH	I	325	0.9	10	90357	8.4	(17)
WENSS	I	325	0.9	18	229420	17	(18)
MIYUN	I	232	3.8	100	34426	40	(19)
4C	I	178	11.5	2000	4844	53	(20)
3CR	I	178	6.0	5000	327	31	(21)
3C	I	159	10.0	7000	470	3.4	(22)

(1) Patnaik et al. (1998); Browne et al. (1998); Wilkinson et al. (1998)
(2) Gregory et al. (1996)
(3) Gregory & Condon (1991)
(4) Becker et al. (1991)
(5) Wright et al. (1994; 1996); Griffith et al. (1994; 1995)
(6) Bennett et al. (1986); Langston et al. (1990); Griffith et al. (1990; 1991)
(7) Otrupcek & Wright (1991)
(8) Fürst et al. (1990)
(9) White et al. (1998)
(10) Condon et al. (1998)
(11) White & Becker (1992)
(12) Mauch et al. (2003)
(13) Colla et al. (1970; 1972; 1973); Fanti et al. (1974)
(14) Ficarra et al. (1985)
(15) Douglas et al. (1996)
(16) Large et al. (1991)
(17) de Breuck et al. (1997)
(18) Rengelink et al. (1997)
(19) Zhang et al. (1997)
(20) Pilkington & Scott (1965); Gower et al. (1967)
(21) Bennet (1962)
(22) Edge et al. (1959)

512 MB RAM. It produces a variable ASCII output that can adapted easily to the purpose of the user. A special output to enter the cross-identifiations into Simbad has been developped. The code has been optimized to find a maximum number of spectra with a relatively small number of misidentifications. It represents thus a promising tool to extract radio spectra from a large sample of radio catalogues stored in VizieR. For the moment there are only 22 catalogue entries in SPECFIND. This number will be increased further in the future. The advantage of this procedure is that radio soectra and cross-identifications are established in advance and can be stored in a 'master list' within VIZIER allowing a fast access.

References

[1] Andernach H. 1999, in:Internet Resources for Professional Astronomy, Lecture Notes of a course given at IXth Canary Islands Winter School of Astrophysics, "Astrophysics with Large Databases in the Internet Age", Nov. 1997, eds. M. Kidger, I. Perez-Fournon, & F. Sanchez, Cambridge Univ. Press, ISBN 0 521 66308 3, p. 67-130

[2] Barnes J. & Hut P. 1986, ApJ, Nature, 324,446

[3] Becker R.H., White R.L., & Edwards A.L. 1991, ApJS, 75, 1 (BWE)

[4] Bennett C.L., Lawrence C.R., Burke B.F., Hewitt J.N., & Mahoney J. 1986, ApJS, 61, 1 (MITG)

[5] Browne I.W.A., Patnaik A.R., Wilkinson P.N., & Wrobel J.M. 1998, MNRAS, 293, 257 (JVAS)

[6] Colla G., Fanti C., Fanti R. et al. 1970, A&AS, 1, 281 (B2)

[7] Colla G., Fanti C., Fanti R. et al. 1972, A&AS, 7, 1 (B2)

[8] Colla G., Fanti C., Fanti R. et al. 1973, A&AS, 11, 291 (B2)

[9] Condon J.J., Cotton W.D., Greisen E.W. et al. 1998, AJ, 115, 1693 (NVSS)

[10] Drinkwater M.J., Webster R.L., Francis P.J. et al. 1997, MNRAS, 284, 85

[11] Bennett A.S. 1962, MRAS, 68, 163 (3CR)

[12] De Breuck C., van Breugel W., Rottgering H.J.A., & Miley G. 2000, A&AS, 143, 303

[13] de Breuck C., Tang Y., de Bruyn A.G., Rottgering H., & van Breugel W. 2002, A&A, 394, 59 (WISH)

[14] de Bruyn G., Miley G., Rengelink R. et al. 1998, WENSS Collaboration NFRA/ASTRON and Leiden Observatory (WENSS)

[15] Douglas J.N., Bash F.N., Bozyan F.A., Torrence G.W., & Wolfe C. 1996, AJ, 111, 1945 (TXS)

[16] Edge D.O., Shakeshaft J.R., McAdam W.B., Baldwin J.E., & Archer S. 1959, MRAS, 68, 37 (3C)

[17] Fanti C., Fanti R., Ficarra A., & Padrielli L. 1974, A&AS, 18, 147 (B2)

[18] Ficarra A., Grueff G., & Tomassetti G. 1985, A&AS, 59, 255 (B3)

[19] Forkert, T. & Altschuler, D. R. 1987, A&AS, 70 77

[20] Fürst E., Reich W., Reich P., & Reif K. 1990, A&AS, 85, 805 (F3R)

[21] Gower J.F.R., Scott P.F., & Wills D. MRAS, 71, 49 (4C)

[22] Gregory P.C. & Condon J.J. 1991, ApJS, 75, 1011 (87GB)

[23] Gregory P.C., Scott W.K., Douglas K., & Condon J.J. 1996, ApJS, 103, 427 (GB6)

[24] Griffith M., Langston G., Heflin M. et al. 1990, ApJS, 74, 129 (MITG)

[25] Griffith M., Langston G., Heflin M., Conner S., & Burke B. 1991, ApJS, 75, 801 (MITG)

[26] Griffith M.R., Wright A.E., Burke B.F., & Ekers R.D. 1994, ApJS, 90, 179 (PMNT)

[27] Griffith M.R., Wright A.E., Burke B.F., & Ekers R.D. 1995, ApJS, 97, 347 (PMNE)

[28] Kulkarni V.K., Mantovani F., & Pauliny-Toth I.I. 1990, A&AS, 82, 41

[29] Kühr H., Witzel A., Pauliny-Toth I.I.K., & Nauber U. 1981, A&AS, 45, 367

[30] Langston G.I., Heflin M.B., Conner S.R. et al. 1990, ApJS, 72, 621 (MITG)

[31] Large M.I., Cram L.E., & Brugess A.M. 1991, The Observatory, 111, 72

[32] Lorimer D.R., Yates J.A., Lyne A.G., & Gould D.M. 1995, MNRAS, 273, 411

[33] Maron O., Kijak J., Kramer M., & Wielebinski R. 2000, A&AS, 147, 195

[34] Mauch T., Murphy T., Buttery H.J. et al. 2003, MNRAS, 342, 1117

[35] Patnaik A.R., Browne I.W.A., Wilkinson P.N., & Wrobel J.M. 1992, MNRAS, 254, 655 (JVAS)

[36] Pilkington J.D.H. & Scott P.F. 1965, MRAS, 69, 183 (4C)

[37] Press W.H, Teukolsky S.A., & Vetterling W.T. 2002, Numerical Recipes in C, Cambridge University Press

[38] Quiniento Z.M. & Cersosimo J.C. 1993, A&AS, 97, 435

[39] Roettgering H.J.A., Lacy M., Miley G.K., Chambers K.C., & Saunders R. 1994, A&AS, 108, 79

[40] Véron-Cetty M.-P. & Véron P. 2003, A&A, 412, 399

[41] Vigotti M., Grueff G., Perley R., Clark B.G., & Bridle A.H. 1989, AJ, 98, 419

[42] White G.L., Batty M.J., Bunton J.D., Brown D.R., & Corben J.B. 1987, MNRAS, 227, 705

[43] White G.L., Batty M.J., Bunton J.D., Brown D.R., & Corben J.B. 1991, MNRAS, 248, 398

[44] White R.L., Becker R.H. 1992, ApJS, 79, 331

[45] White R.L., Becker R.H., Helfand D.J., & Gregg M.D. 1998, ApJ, 475, 479 (FIRST)

[46] Wilkinson P.N., Browne I.W.A., Patnaik A.R., Wrobel J.M., & Sorathia B. 1998, MNRAS, 300, 790 (JVAS)

[47] Wright A.E. & Otrupcek R. 1990, Australia Telescope National Facility, Parkes (PKS)

[48] Wright A.E., Griffith M.R., Burke B.F., & Ekers R.D. 1994, ApJS, 91, 111 (PMNS)

[49] Wright A.E., Griffith M.R., Hunt A.J. et al. 1996, ApJS, 103, 145 (PMNZ)

[50] Zhang X., Zheng Y., Chen H. et al. 1997, A&AS, 121, 59 (MIYUN)

Appendix 1

Representative sample of identified radio spectra (appendix1.ps).

Appendix 2

Representative sample of source, for which no spectrum could be found (appendix2.ps).

Appendix 3

Sources with spectral index $b < -2$ (appendix3.ps).

Appendix 4

Sources with spectral index $b > 1$ (appendix4.ps).