Chapter 2

Observations and Software system for data analysis

"In God we trust,
all others bring data."
- W. Edwards Deming

"You think - you know when you can learn,
are more sure when you can write,
even more when you can teach,
but certain when you can program."
- Alan Perlis

In this chapter, we summarize the observations carried out for the survey (prior to this thesis work) followed by organization of the data set. In the second part we discuss additions to provide various new functionalities and improvements on the existing software system for data analysis, to make it complete and robust.

2.1 Observations

One of the disadvantages of an aperture synthesis telescope like MRT is the long span of time over which the observations have to be made to record the visibilities for synthesizing the final images. The observations for the survey were carried out during the period May, 1994 to March, 1999. In this section, we describe the mode of observations, the various cycles in which the observations were carried out and the resulting survey data set.

2.1.1 Mode of observations

At MRT, all spacings available in a square aperture are obtained by correlating the EW arm of the $\top$ with the NS groups. One of the advantages of a $\top$ shaped array like MRT is that it samples the visibilities on a uniform regular rectangular grid and not quasi randomly like a Y array or regularly on great circles facing the poles like Cambridge Low Frequency Synthesis Telescope (CLFST). These instruments require the measured visibilities to be resampled onto a regular grid and this operation introduces yet another sampling function before the visibilities are transformed to an image. The corrections due to irregular weighting and their relative contributions based on distance of these samples from the grid centers are not required in our case. This also makes the deconvolution relatively simpler.

Our aim is to sample the visibilities with NS baselines from 0 m all the way up to 880 m, every 1 m ($\frac{\lambda}{2}$ at 150 MHz), which keeps the grating response at the horizon while observing at 150 MHz. Even though the frequency of operation was shifted to 151.5 MHz, this sampling still keeps the grating response well outside the main beam of the primary beam response. A 512 channel complex correlator is used to measure the visibility function by correlating 32 EW groups with 16 NS groups (15NS and E16). E16 as shown in Fig. 1.6 is the last group of the EW array and is fed into the correlator in place of 16^{th} NS trolley. This gives a set of baselines between E16 and the EW array on all the observing days which helps to check the repeatability of the data. The baselines with each NS trolley gives 31 independent closure information which can be used in calibration. At the same time this reduces the number of usable NS trolleys to 15.

MRT measures different Fourier components of the brightness distribution of the sky in 63 different sets of configurations of the trolleys in the NS arm over a period of time in

Block number	Allocation range	Baseline covered (along NS)	Number of delay settings
block-1	1-6	6 m-94 m	1
block-2	7-12	2 m-5 m, 95 m-171 m	1
block-3	13-18	172 m-261 m	2
block-4	19-24	262 m-351 m	3
block-5	25-30	352 m-441 m	4
block-6	31-36	442 m-531 m	4
block-7	37-42	532 m-621 m	4
block-8	43-48	622 m-711 m	4
block-9	49-54	712 m-801 m	4
block-10	55-63	796 m-880 m	4

Table 2.1: The observations over the 880 m NS track are divided into 10 blocks each covering about 90 m. The number of delay settings used during observations for each block is also shown.

order to sample all baselines in the NS arm every 1 m. Each configuration corresponds to a given placement of the NS trolleys and is referred to as an *allocation*. Due to local terrain the trolleys in the southern part of the NS arm cannot approach the EW arm closer than 11 m. So the visibilities with trolley distance from 2 m to 11 m along the NS from the EW arm are sampled using a trolley on the northern extension. The trolleys cannot be placed closer than 2 m due to the physical size of the trolley itself. In each allocation (barring when a trolley needs to be located on a forest road and a few exceptions when observations on certain baselines are repeated), the 15 NS trolleys are spread over 84 m with an inter-trolley spacing of 6 m as shown in Fig. 1.6. The 6 m distance between each NS trolley also minimizes shadowing of one trolley by another.

In each allocation observations are carried out for a minimum period of three days (Sachdev & Udaya Shankar, 2001b; Sachdev, 1999). This ensures that we have sufficient good data for imaging even if one has to discard data due to several factors such as man made interference, Sun affecting the observations (increased solar activity) and unexpected instrumental failures. In the next allocation the trolleys are moved by one metre each. After 6 consecutive allocations, we get visibilities measured over 90 m stretch, sampled every 1 m. We refer to the measurements made over 6 consecutive allocations filling up a 90 m stretch along the NS track as a *block*. After measuring the visibilities in one block all the 15 NS trolleys are moved to cover the next 90 m stretch corresponding to the next block. The 63 required allocations are obtained in ten blocks, where the tenth block includes 3 additional allocations from 61 to 63. These are used to measure certain baselines which could not be measured in the tenth block due to local terrain constraints. The visibilities are measured with different delay settings to minimize the effect of bandwidth decorrelation. With each delay setting, a small part of the sky in declination can be observed without appreciable decorrelation (<20%) which is referred to as a delay

Cycle No.	Allocations covered	Observation period Start	End	Sampling frequency (MHz)	Mode	Data collected (hours)
1	1-12	July 94	Sep 94	12	EW×NS	835
2	13-24	Sep 95	Dec 95	2.65625	EW×NS	1469
3	1-25; 1-63	Jan 96	Mar 99	2.65625	EW×NS E×EW, NS×(NS+W)	17320
Total	1-63	July 94	Mar 99			19624

Table 2.2: The various cycles during which the observations were carried out for the survey. Only the data files which are complete for one sidereal hour range have been considered. For observations with allocations 1 to 12, each visibility value is stored as **int** (occupies four bytes) while for observations with allocations 13-63, each visibility value is stored as **short** (occupies two bytes).

zone (Sachdev & Udaya Shankar, 2001b). The number of delay settings used for a block vary from one to four and depends upon the maximum length of the NS baseline covered in that block. The positions covered and the number of delay settings used in each block are given in Table 2.1.

Since observations are carried out for three days in each of the allocations, it takes about six months to obtain data for 63 allocations. However, in this period the Sun moves through half the sky (12 hours in RA), thereby preventing the full sky coverage with six months data. We therefore carry out the observations in two rounds. In the second round we repeat the observations with the same allocations after a gap of about six months. This ensures that we have night time observations covering all sidereal hours for each allocation. In addition there have been repetitions in the observations, as the array was being commissioned and tested in various stages while the observations were going on.

2.1.2 Observing cycles

The entire observations carried out for the survey can be broadly divided into three major cycles, as shown in Table 2.2. In the first cycle observations were carried out to cover all NS baselines from 2-171 m (i.e. blocks 1 and 2). Data observed in allocations 1 to 12 (i.e. block 1 and 2) is also referred to as Short Base Lines data[1] as on these baselines the bandwidth decorrelation is not significant over the range of declination covered by MRT. Hence it is adequate to measure the visibilities with only one delay setting. The sampling used is 12 MHz and the integration time is ≈ 1.095 sidereal seconds.

The second cycle of observations started after an years gap, using the newly built recirculator (Sachdev & Udaya Shankar, 2001b; Sachdev, 1999). During this cycle, the NS base-

[1]Data observed in allocations from 1 to 12 is referred to as Short Base Lines (SBL) data while data observed in allocations 13 to 63 is referred to as Long Base Lines (LBL) data.

lines from 172-351 m (i.e. blocks 3 and 4) were covered. The visibilities for both the blocks were measured with different delay settings as given in Table 2.1. The sampling frequency used is 2.65625 MHz and the integration time is $\approx$ 1.088 sidereal seconds.

In a $\top$ array like MRT there are no redundant baselines in its usage for image synthesis in pencil beam mode ((E+W)×NS) but since EW and NS arrays are highly redundant arrays with equally spaced groups, there are some baselines which occur more than once. In the third cycle the correlation receiver was configured to correlate the E×(E+W) and NS×(NS+W) in addition to the pencil beam mode, which can be used to independently calibrate the EW and the NS arrays using redundant baseline calibration (Hamaker et. al., 1977). Only a small fraction of integration period was used for the measurements with E×(E+W) and NS×(NS+W) mode (equivalent to ≈0.05 s). During this cycle, first the remaining allocations from 25 to 63 were covered and subsequently observations for all the 63 allocations were repeated. The visibilities covering all the blocks were measured with different delay settings as given in Table 2.1.

The Data Set: The MRT survey dataset comprises of ≈20,000 hours of astronomical observations with a total size of ≈ 1 TB. To facilitate data handling and data processing, it has been organized in hourly Local Sidereal Time (LST) files. Each data file contains visibilities measured with one delay setting for one sidereal hour duration. For the third cycle of observations, visibilities measured in each of the (E+W)×E and (NS+W)×NS configuration are also recorded separately for each sidereal hour duration. The data files are in binary and use the C unformatted style. For Short Base Lines data, each visibility value is stored as **int** (occupies four bytes). For Long Base Lines data, each visibility value is stored as **short** (occupies two bytes). The nomenclature of the data file is such that most of the preliminary information about the data file like Julian Day (JD) of observation, allocation, sidereal hour duration, delay zone etc. can be inferred from the name of the file itself.

2.2 Data organization

The data was stored as and when the observations were carried out in chronological order of observations on 160 data cartridges (Exabyte make) of native capacity 5 GB each, after compression using lossless compression utilities. The imaging at MRT is carried out on sidereal hour basis. To image one sidereal hour and full declination range of MRT, one has to retrieve data spread across several data cartridges. Data retrieval from data cartridges is sequential and extremely slow. Due to this, data retrieval was one of the most time consuming step in data processing.

During the last 3-4 years, the dramatic improvement in technology and severe compe-

tition in computer industry has led to significant increase in the capacity of hard drives. Due to easy availability of high capacity hard-disks (capacities exceeding 100 GB) at low costs it has become practically feasible to achieve data processing setup with several TB of memory on hard disks. Taking advantage of this we have now built up a database during the course of this dissertation in which the entire MRT survey data is available (in read only form) on networked hard drives accessible to the computers used for MRT data processing work at Raman Research Institute (RRI).

In order to implement any automatic scheme for data processing, the corrupted data needs to be identified and should be avoided during imaging. Data corruption can occur due to erroneous recording, improper compression and archiving or due to the data cartridges going bad. Each data file was subjected to data integrity checks for memory, header positions and sidereal time stamps. Software was developed in order to identify and remove the corrupted data and duplicate data files from the database. About 4% of the data files were found to be corrupted. Data organization has facilitated immediate data access, helped in streamlining the data processing and has brought down the time spent on data retrieval to a minimum. Since the imaging is done on sidereal hour basis, we have also stored data on hard disks on the basis of sidereal hour and within each sidereal hour range in order of increasing allocation. A 3-tier archive based on Mammoth-2 data cartridges of native capacity 60 GB each, DDS-3 data cartridges of native capacity 12 GB each and on hard drives of capacity 250 GB each, have been maintained for any inadvertent exigencies. A copy of this has been also made available to the observatory in Mauritius. This integrity checked database contains a total of about 80,000 files of hourly visibility data and is used while imaging for the survey.

The available data for the survey: A simple analysis was carried out on the data to ascertain, if we have enough data for completing the survey. As imaging is done on sidereal hour basis, only data files containing visibilities for one full sidereal hour range were considered. Since night time observations are generally found to be interference free and also have minimum interference from the Sun, it is classified as class-I data. Class-I refers to a data file observed when the Sun's RA is 6 hours away from the LST of observation. On similar lines class-II refers to a data recorded when the Sun's RA is 3 to 6 hours away from the LST. Similarly we have class-III and class-IV data which refer to data recorded when the Sun's RA is 2 to 3 hours and less than 2 hours away from LST respectively. Fig. 2.1 shows the availability of class-I data file for different allocations and sidereal hours. It is clear that there is at least one Class-I data file for most of the allocations for all the sidereal hours. There are very few gaps which can be easily filled in by the Class-II data. The analysis indicates that the data set for the survey is indeed complete.

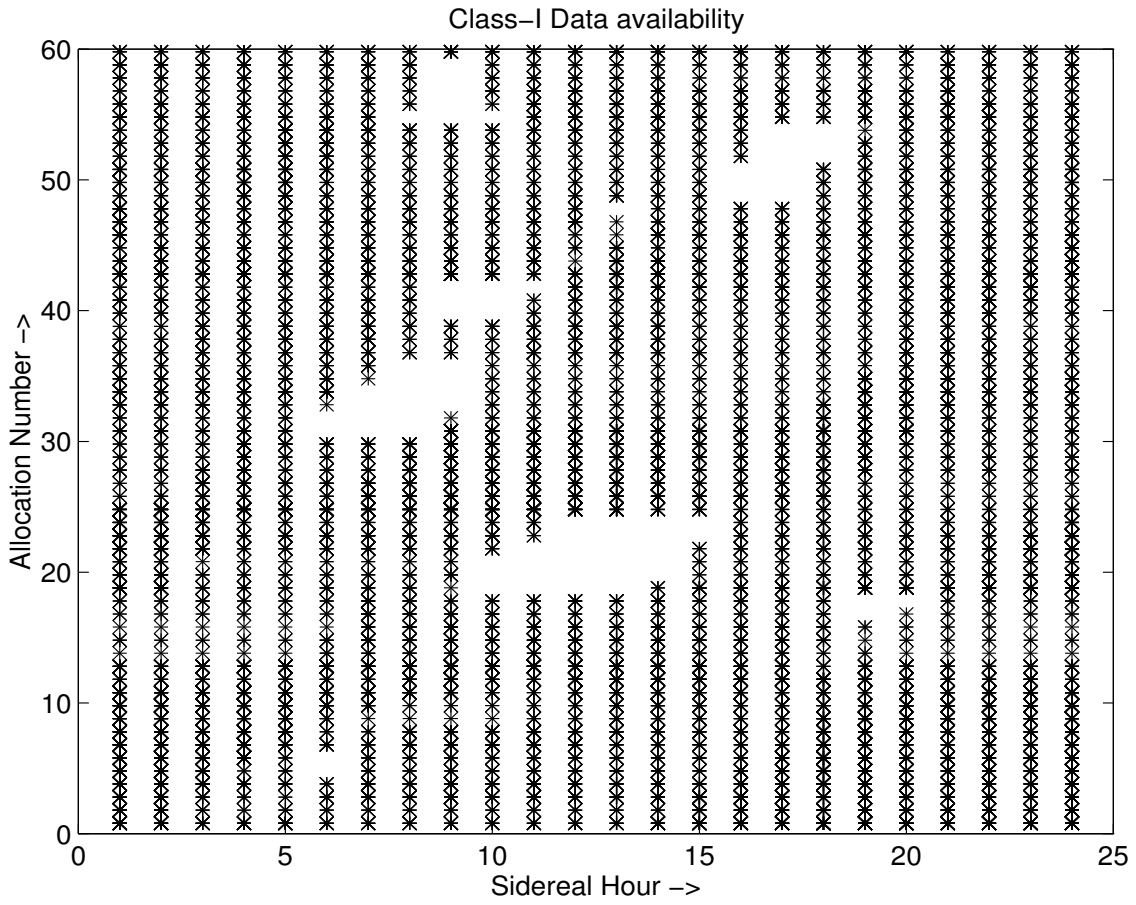

Fig. 2.1: The figure shows the availability of Class-I data for different allocations and sidereal hours (denoted by a *). There are a few gaps which can easily be filled by Class-II data.

2.3 Software system for data analysis

In this section we briefly describe the data analysis software system developed in-house for debugging and wide field imaging with the non-coplanar MRT array. An hierarchical software system based on a mix of top-down and bottom-up approach consisting of separate application programs with specific functionalities (top-most layer) and using general purpose generic libraries (lowest layer) was developed to accomplish this purpose. Algorithm development is a separate vital discipline and we would not be including this in our discussions as a part of software development in this chapter. The development of the data analysis software system can be broadly divided into two stages. The first version was developed before the commencement of this thesis work and the second one with substantial additions and improvements has been developed during the course of this dissertation. The former is referred to as MARMOSAT[2] while the latter is referred to as eXtended MARMOSAT (X-MARMOSAT). We would first briefly mention their main objectives indi-

[2] **MA**u**R**itius **M**inimum **O**perating **S**ystem for **A**rray **T**elescopes

vidually and then discuss the design aspects of the overall software system.

2.3.1 MARMOSAT

MRT is in a region of rough terrain due to which it is non-coplanar and also the alignment of arms is far from perfect. Many if not most of the difficulties which have to be solved would not have arisen in the first place if a better site had been available. In its configuration, the method of observation and data analysis, MRT cannot be categorized as one belonging to the telescopes else where in the world. This partly compelled the in-house development of a software system for imaging with the MRT. The development of software started as the correlator was put through the first tests. The main objectives of MARMOSAT suite which was mainly developed by R. Dodson (Dodson, 1997) are two fold. One of it provides a link with the correlator system while the other provides off-line data analysis capabilities. The data processing steps implemented in MARMOSAT are shown in Fig. 2.2.

The first part uses the correlator output for online monitoring of the health of the array (amplitude and phase of various baselines, antennae based amplitude and phase, system temperature variation with time, *cosine* and *sine* correlations etc.). Two programs *Online* and *Health* were developed to perform system checks in real time on the stream of data being received continuously, to quickly detect gross errors like malfunctioning components. *Online* displays the visibility data on a few baselines in detail while *Health* gives the aggregate properties of several baselines. They perform numerous tasks on the data and provide options to check various array parameters. The data is stored temporarily on hard disk and later on data cartridge tapes via a distributed network.

The second part carries out off-line processing of the measured visibilities. The most important of these include interference detection and excision, calibration and transforming the visibilities to brightness distribution taking into account the non-coplanarity of the array. A detailed discussion about the software aspects of these is beyond the scope of this dissertation and can be found in Dodson (1997).

2.3.2 eXtended MARMOSAT

The eXtended MARMOSAT comprises of various new functionalities and improvements which were incorporated on the existing MARMOSAT to make the data analysis software complete and robust during the course of this dissertation. Fig. 2.3 shows the main steps of data processing implemented in the overall data analysis software system starting from recorded visibilities up to the source catalogue construction.

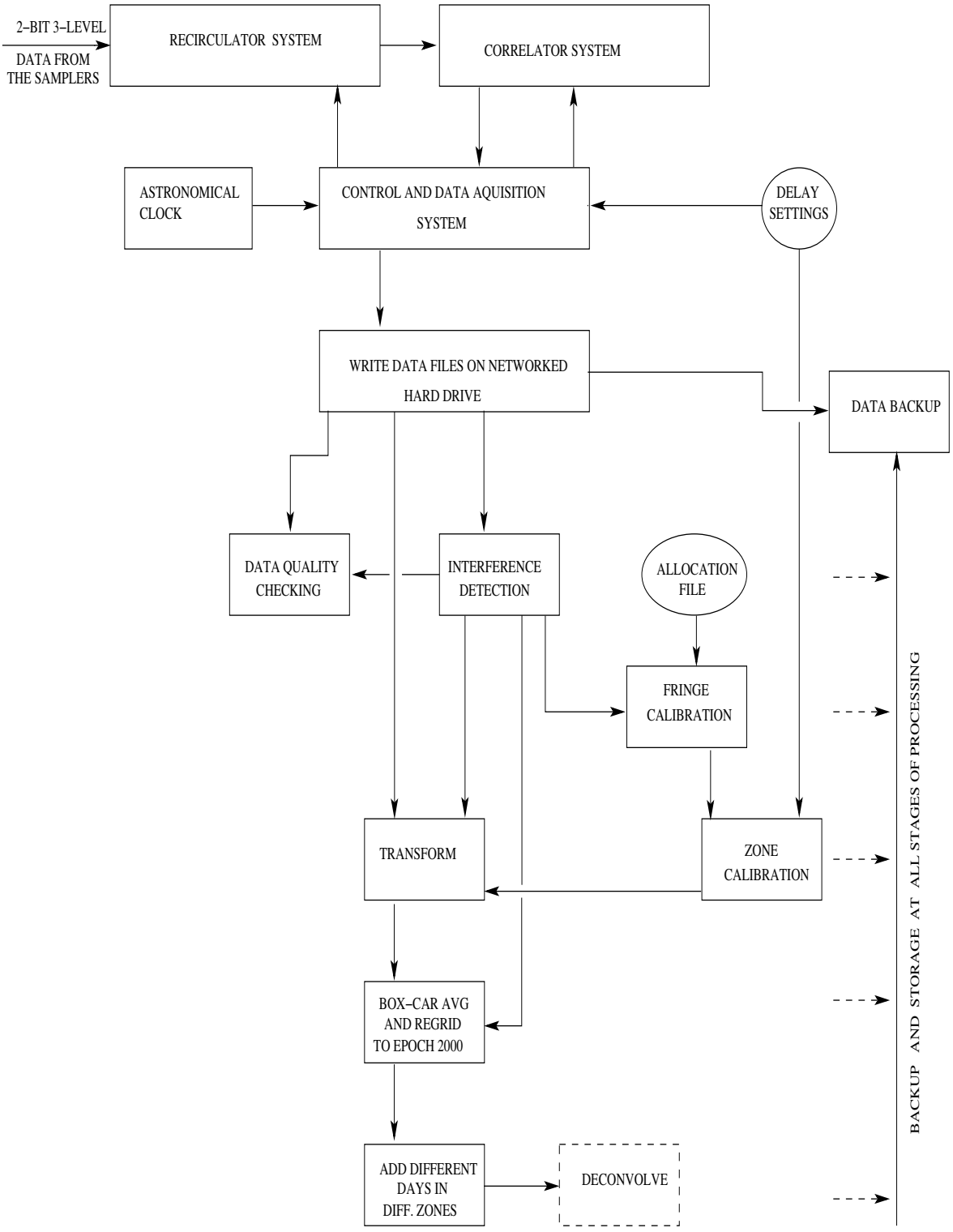

Fig. 2.2: A schematic showing data processing steps to obtain full resolution dirty images using MARMOSAT which was mainly developed by R. Dodson (courtesy Sachdev (1999)).

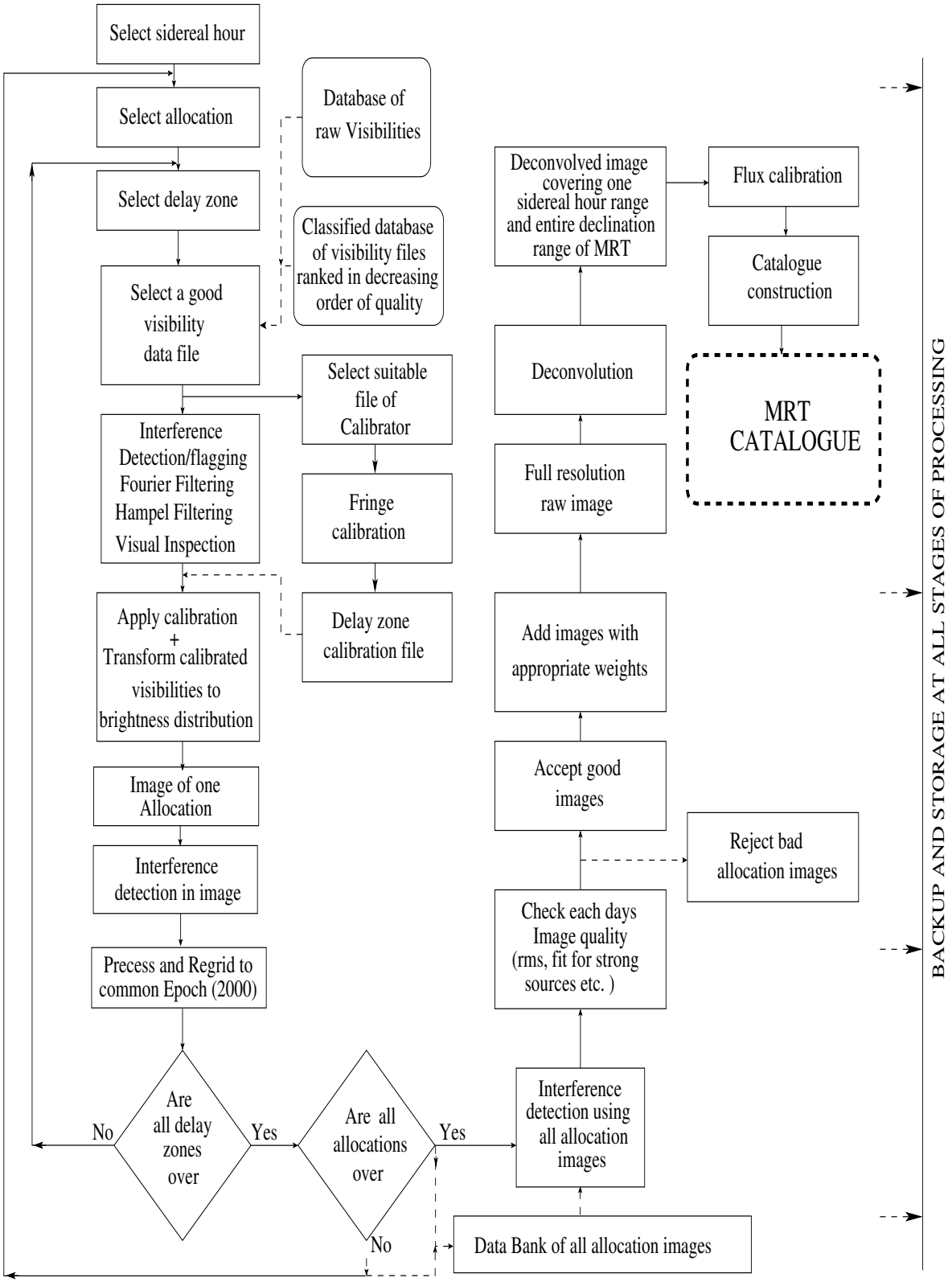

Fig. 2.3: A schematic showing the main steps of data processing implemented in the overall data analysis software system to synthesize full resolution deconvolved image including source catalogue.

Additional Functionalities : The main objectives of the additional functionalities which were added to the data analysis software include (a) Automatic evaluation of data quality (b) A hierarchical RFI mitigation system (c) Image analysis at the level of each day and full resolution images[3] (d) Deconvolution of wide field images with a non-coplanar array (e) Flux calibration (f) Source extraction and catalogue construction from the deconvolved images (g) Correcting for the errors in time stamping in the data files.

To accomplish most of these additional functionalities, a number of different programs were developed (for each functionality often at multiple stages of data processing). All these required sophisticated efficient software including the need for browsing large amount of data with interactive Graphical User Interface (GUI). The software developed to provide additional functionalities in X-MARMOSAT is fairly large and complex. It comprises of more than 60,000 lines of code in a variety of languages namely **C, Perl, Matlab** and **F77** and in addition uses shell scripts extensively. Detailed discussion on the software aspect would be excluded in this thesis keeping in mind that the thesis deals with astronomy *per se*. The algorithms would be discussed briefly as and when required in the later parts of this thesis.

An important point worth mentioning here is that we carry out detection of interference in self correlations, increase their integration time and correct for the errors in time stamps in the data files before any other steps of data processing are carried out (See Sec. 4.3.4 and Sec. 5.1 for details). The original data is not modified at any stage and the interference excised and smoothed self correlations along with corrected sidereal times are stored in a separate binary file corresponding to each data file as they are used by most of the programs for data processing. Since all the existing programs in MARMOSAT were designed to access the self correlations and sidereal time from the original visibility file only, we needed to carry out appropriate changes in order for the programs to access the values stored separately in the newly created file instead. This required us to read the source code of each of the application program and the libraries in the existing MARMOSAT (only those parts which are used for off-line data analysis), to incorporate the necessary changes.

In order to ensure that no previously working functionalities have failed and that the new modifications to the program have not inadvertently introduced errors, verification/regression testing was carried out. In this, each program was executed with different sets of input parameters, before and after modifications and the outputs at different stages were compared. In addition, the expected values of variables were compared at intermediate stages using a debugger for both sets of programs. Parts of the program which were

[3]Images for a survey with partial resolution ($17'\times23'$) with MRT were made earlier by Golap (1998) while data was being collected with longer NS baselines. Due to this the term full resolution image is used and refers to an image with a resolution of $4'\times4\farcs6\sec(\delta+20\fdg14)$ made using data from all the allocations.

dead codes and never executed were retained as it is in order to keep the changes from the original to a minimum.

Improvements : The improvements were carried out only in those programs of the MARMOSAT which deal with the off-line data analysis and are briefly described below.

Software was developed in order to identify and remove the corrupted data and duplicate data files from the database as a part of data organization described in Sec. 2.2. The most computationally intensive step during imaging is the inversion of the calibrated visibilities into images, which requires use of Direct Fourier Transform (DFT) due to non-coplanarity of the array. The run time performance for the inversion of visibilities was successfully improved by use of look-up tables, by a factor of ten. The existing programs were optimized by avoiding redundancy and reducing loop overhead by reducing the number of iterations and replicating the body of the loop (loop unrolling).

We incorporated appropriate changes to improve the flexibility of usage of a few programs, like ability to run them on data of different cycles. The programs were written originally for Digital Unix O/S (Dec-Alpha platform), Linux (IBM PC) and for Digital Unix (Sun Platform). Our aim was to bring in conformity the entire existing software to a Linux platform which is being used both at RRI and Observatory at Mauritius, for data processing work. While most of the programs could be easily compiled on Linux, a few of them needed modifications. The changes include replacing by newer syntax, the old subroutine calls which had become obsolete in the newer versions of the Linux operating system and the software packages on which the programs depend. All the programs in the data analysis software system were compiled with the warning flags on, to check for any suspected errors related to improper memory allocation, exchange of data to functions etc..

2.3.3 Software design

The design of data analysis software system is based on modular approach. The focus has been to choose schemes and methods which are natural and appropriate depending upon our particular strengths and problems. The data analysis software system consists of stand-alone programs which have to be executed one after another in a sequential manner. The programs communicate via standard data files rather than having all possible operations integrated into one huge kernel program. This is similar to the approach taken by Unix and Linux Operating Systems. This facilitates easy maintenance, easier addition of new functions or utilities and flexibility for up-gradation. In this approach it is difficult to run the programs blind which ensures that the processing is carried out carefully. The user interface is provided by using external display softwares. The programs which are executed in later stages do require the output of earlier programs as inputs along with a

variety of flag options which provide flexibility. The disadvantage is that the tasks often require scripts to chain together several programs for their execution. This is important in situations when one has to process a large number of files as in case of MRT. Shell scripts are extensively used in batch processing mode for imaging.

In view of the large size and complexity of the software requirement for the eXtended MARMOSAT, software design and careful planning was essential to ensure that it works correctly, can be maintained and extended in future. Now we discuss the salient features behind its design.

Software design is the process of defining the architecture, components[4], interfaces and other characteristics of the system and the results of that process. In this, we analyse the software requirements to produce a design of internal structure and organization of the system which serves as a basis for its construction while adhering to general principles of good software design. This involves the task of division of the system into subsystem and components and how these will be connected including their interfaces. During the software design, one faces a series of design issues and each issue normally has several alternate design options. In order to make a design decision in the design space one uses knowledge of the requirements, the design created so far, the available resources, software design principles and best practices based on one's experience of what has worked well in the past. Here the main principles and the basic guidelines adhered behind the design of the developed software for eXtended MARMOSAT would be briefly mentioned.

- **Top-down and Bottom-up design :** We first designed the very high level structure of the system without considering the implementation details to evolve with a good system structure. Later, we started with the low level utilities keeping in mind their re-usability to construct the high level structure. The lowest level components were designed to behave as a single logical entity. Subroutines/Functions which perform a variety of tasks with very generalized usage were avoided to the maximum possible extent. Before implementing a particular algorithm, separate flowcharts emphasizing the top-down and bottom-up design were produced. A piece by piece comparison of the flowcharts was used to analyse the requirements and arrive at a fair balance between these two approaches in order to give a software system a good structure and to ensure that the reusable components can be maximally exploited.
- **Maximize Cohesion :** Cohesion[5] is a measure of the strength of functional related-

[4]Component : Any piece of software which has a clear role. It may be isolated allowing one to replace it with a different component that has equivalent functionality.

[5]Due to wide popularity of object oriented programming and reusable code engineering, cohesion and coupling are sometimes associated to be paradigms of object oriented programming itself; nothing could be farther from truth. Cohesion and coupling grew out of structured programming research in 1970's and

ness of elements within a module i.e. how well the lines of a source code within a module work together to provide the required functionality. In software highly cohesive programs are desired in order to make the system as a whole easier to understand. During the software development we grouped parts together based on three basic criteria : (a) data on which a module operates (b) logical relationship and (c) output of one part being used as input of another. The basic philosophy while designing the modules was to ensure that the higher level modules control the overall logic and the lower level modules do the nitty gritty work.

- **Minimize Coupling :** Coupling occurs when there are interdependencies between one component and another. Changes in one place require changes somewhere else which make it hard to interpret a particular components overall behavior. For a well designed software low coupling is preferred. During the software development while passing the information which needs to be communicated between modules, data was always passed using explicitly stated argument or parameter lists and never shared globally except in cases where it was absolutely necessary. Logical dependencies of content were avoided by ensuring that at no stage a module refers to inside another module in anyway. The control at any stage in the programs developed always returns to the calling location and sudden jumps from one part to another were avoided. Functions were always declared in full prototype mode before they can be used. Data structures were cleared once they were not required to free the memory and ensure they cannot be used later inadvertently.
- **Easy viewing of data :** It should be easy to display and view the data and the results of data processing. Since it is difficult to write a data display software which satisfies all present and future needs, a good approach is to design a data analysis software which uses external stand-alone display software. This reduces the effort for development of display software which requires different specialized software skills. There are large number of freely available external graphic display packages which were suitable for our purpose. These include pgplot, gnuplot, octave, scilab etc.. They were chosen depending upon their capabilities, the ease with which they can be used for the desired application and their compatibility with the language used.
- **Simplicity :** The software was kept as simple as possible. It is easy to use tricks just to make the code apparently shorter but it may become difficult to understand later by the developer and user both. We employed short cut tricks only when it did serve a substantial purpose like significant decrease in the computation time.

were discussed by Larry Constantine and Edward Yourdon in their 1976 landmark book, 'Structured Design', subsequently their software quality metric (measure) was coined by Larry Constantine.

- **Programming languages :** In principle, all the requirements can be carried out using a single programming language but it generally results in a lot of complicated and inefficient code as each language has its own strengths and weaknesses. In view of this, the language was chosen depending upon the requirements.

  Most of the computationally intensive programs are generally written in ANSI-C with very few exceptions. The ANSI-C was used so as the programs are portable as the processing is carried out both at RRI and observatory at Mauritius. C was preferred over F77 due to its better memory management and string parsing capabilities, availability of an inbuilt debugger (apart from many external ones) and comparable computational speed. A few F77 subroutines mostly taken from Numerical Recipes are called from the C programs as and when required.

  Data processing steps which required a fair combination of scripting, text processing, display and computation were carried out using Perl programming language. Perl (Larry et. al., 2000; Srinivasan, 1997) was chosen due to its flexibility, ease of programming, excellent text parsing capabilities, compatibility with large (nearly all) number of graphical display packages, speed, free availability and inbuilt debugger. It provides all the benefits of an interpreted language while coding and debugging and the speed of a complied language during run time. It is Object Oriented and a cross platform language which is portable across various operating systems. It supports referencing but does not directly support addressable pointers. This enables easy construction of complex data structures without dangers inherent in pointer arithmetic. Also its standard library comes with comprehensive eXtensible Markup Language (XML) support and Graphical User Interface (GUI) apart from long list of support modules. Perl has provision for embedding C code inside a Perl program and one can also embed a Perl code inside a C program with support of modules, which makes it useful for MRT where most of the software has been written in C.

  Matlab was used mainly for image analysis. It has a very user friendly interface for interactive command operations and display. Most of the analysis can be carried out using a large number of preprogrammed functions available for matrix manipulation. Codes can be written very quickly and debugged using the inbuilt debugger.

- **Documentation :** Appropriate documentation is provided at each stage. The documentation is generally built in the programs. This helps the software and the documentation to be in synchronization. The general purpose help can be obtained by invoking the -help flag while executing. In order to keep track of the various output files produced by different programs, an ASCII header is used for each file by the application programs to read or transfer the information from each other. To maintain

a simple standard tracking system, the names are standardized and the software by knowing the name of a file can work out basic information about the file.

- **Libraries :** At the lower layer of software system are libraries. These contain a set of useful functions which are required very often by most of the programs. These can be automatically linked while running the individual application program. Any modification or improvement in these libraries affects across the entire software system. This helps in optimizing the maintenance of the software by keeping most of the common tasks at one place. These also include useful routines available in Numerical Recipes in C (Press et. al., 1989).
- **Header and Miscellaneous files :** Header files contain parameters of the data structure and the array settings during the observation of the data file like integration time, maximum number of correlations etc.. They also contain the definitions of the functions in the libraries and application programs so that they do not have to be declared individually in each application program. A few examples of header files are marmosat.h, complex.h and defvar.h. Other important files are position files which have the antennae coordinates for each allocation, Delay files having the delay settings with which the data was observed, Phase per delay files which contain the equivalent phase changes across each baseline corresponding to a unit delay etc..

In this chapter, we have given a brief overview of the observations carried out and the data analysis software system developed in-house for processing the visibilities. In the next chapter, we discuss an algorithm for automatic scientific classification of the visibility data to select the best data for imaging.