Tracking accuracy of 3.6m Telescope at Devasthal and Temperature map & optical depth of star forming molecular clouds from Planck satellite data

INTERNSHIP PROJECT REPORT

by

NIKHIL S HUBBALLI

(SC13B158)

Department of Earth and Space sciences

Indian Institute of Space Science and Technology

Thiruvananthapuram

June 2016

Tracking accuracy of 3.6m Telescope at Devasthal and Temperature map & optical depth of star forming molecular clouds from Planck satellite data

B.Tech project Report submitted
in partial fulfillment for the award of the Degree of

BACHELOR OF TECHNOLOGY

in

PHYSICAL SCIENCES

by

NIKHIL S HUBBALLI

pursued in

DEPARTMENT OF EARTH AND SPACE SCIENCES

to

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

Thiruvananthapuram

June 2016

BONAFIDE CERTIFICATE

This is to certify that this project report entitled TRACKING ACCURACY OF 3.6M TELESCOPE AT DEVASTHAL AND TEMPERATURE MAP & OPTICAL DEPTH OF STAR FORMING MOLECULAR CLOUDS FROM PLANCK SATELLITE DATA submitted to Indian Institute of Space Science and Technology, Thiruvananthapuram, is a bonafide record of work done by NIKHIL SHANKARAPPA HUBBALLI under my supervision from 7th June 2016 to 18th July 2016.

Signature of the Supervisor

BRIJESH KUMAR
SCIENTIST-E
Name and Designation

Scientist "E"
ARIES, Manora Peak
Nainital.

12-8-16

Countersignature

निदेशक
Director
आर्यभट्ट प्रेक्षण विज्ञान शोध संस्थान (एरीज)
Aryabhatta Research Institute of Observational Sciences (ARIES)
मनोरा पीक, नैनीताल-263 002
Manora Peak, Nainital-263 002

Name, Designation, eg. Head of Dep. / Divisional Head

Declaration

I declare that this report titled "Tracking accuracy of 3.6m Telescope at Devasthal and Temperature map and optical depth of star forming molecular clouds from Planck satellite data" submitted in partial fulfillment of the Degree of "Bachelor of Technology in Physical Sciences" is a record of original internship work carried out by me under the supervision of Dr. Maheswar Gopinathan, and has not formed the basis for the award of any degree, diploma, fellowship or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Nikhil S Hubballi
August 2016

Acknowledgements

First and foremost I thank Dr. Maheswar Gopinathan for providing me this opportunity to work on the study of molecular clouds in the field of Astronomy and Astrophysics. He has been a motivation throughtout the entire duration of this summer internship project. His support and guidance has been invaluable for my work. I also like to thank Brijesh Kumar for his support and guidance in providing telescopic data. I also thank Dr. Resmi Lekshmi for offering me the opportunity to pursue this project in one of the premier institutes. I also thank Shankar Ram for sharing his expertise to help build my work with his insights in the field. I thank my family and friends for providing the encouragement to excel in my goals. Lastly, I would like to thank all my teachers and professors for shaping my career and enabling me to undertake this project.

Name: Nikhil S Hubballi
Roll No. : SC13B158

Abstract

The report consists of two parts. In the first part, an analysis is done on the extent of accuracy while tracking astronomical objects through the 3.6 meter telescope situated at Devasthal, Uttarakhand. It is required to have the tracking accuracy <0.1 arcsec RMS for less than one minute in closed loop for wind inside the dome <3m/s. <0.11 arcsec RMS for less than one hour in closed loop for wind inside the dome <3m/s.

A series of astronomical long exposures are taken at different altitudes and azimuth locations. The images, through Test-camera, are taken on axis, as defined by the centre of rotation of the Cassegrain derotator. The centroid of the stars are determined and displacement with time is noted, and hence tracking error is calculated. The analysis is done on various datasets covering different altitudes and azimuth locations in closed loop. Tracking error plots as well as RMS error over different exposure intervals has been plotted.

In part two, a method to get the temperature map of molecular clouds is worked out using *Planck* data. *Planck* enables unbiased mapping of Galactic emission from the most diffuse regions to the dense regions of molecular clouds in sub-millimetre and millimetre region. The emission spectrum measured by *Planck* and IRAS can be fitted pixel by pixel using a single modified blackbody. There are residuals detected at 353 GHz and 143 GHz. Positive residuals can be seen in 217 GHz and 100 GHz bands in the molecular regions due to ^{12}CO and ^{13}CO emissions. Maps of dust temperature (T), spectral emissivity index (β), and the dust optical depth (τ) are obtained. Temperature map explains the cooling of the dust particles in thermal equilibrium with the incident radiation field, with 16-17 K in diffuse regions to 13-14 K in the dense regions. There is significant anti-correlation between temperature and spectral emissivity index. The dust optical depth map is used to explain the spatial distribution of the column density of the molecular cloud from dense to diffuse regions. Taurus molecular cloud is used to validate the methodology using a reference research on the topic, later the procedure is applied to two more molecular regions, Lambda Ori and Perseus. This methodology can be extended to any molecular cloud from the *Planck* data.

Table of contents

List of figures

List of tables

Part I

Tracking Accuracy of 3.6 meter Telescope at Devasthal

Chapter 1

Introduction

In astronomy, whenever we want to make an observation for a prolonged duration, it's necessary to track the object in the sky keeping it the field of view of the telescope. But, it's not easy as the object keeps moving through the sky with time and tracking it becomes a tedious job manually. Therefore, we install mechanical systems that can track the object based on several factors. Earth's rotation is one such purpose to be compensated during tracking. The Earth's rotation relative to the stars is called a sidereal day, it has a length of 86164 seconds (a Solar day is 86400 seconds). So, obviously the telescope Right Ascension (RA) axis must rotate once in exactly 86164 seconds in order for the observer not to experience stars drifting in the field of view. This is especially important for deep sky photography where long exposures are employed. If there is drift, the stars will be elongated instead of round, or they will leave the field of view alltogether. Even when we go for long exposure photography of the objects (>1min), this drift becomes significant as field of view is narrow.

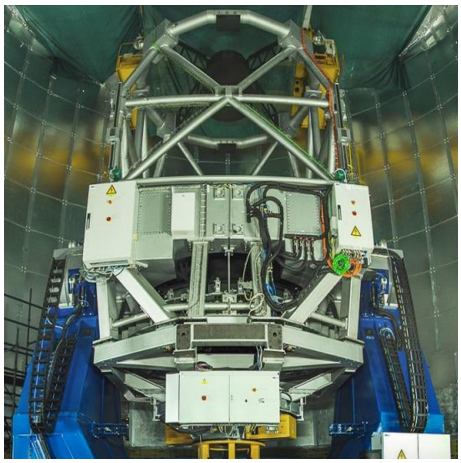


Fig. 1.1 3.6 meter telescope at Devasthal

There are several reasons for this drifting:

1. Inaccurate Polar alignment
2. Incorrect RA motor speed
3. Periodic error in RA

Tracking of objects becomes a necessary task because of the errors that creep in the observations. But, achieving a mechanical system which can track an object in the sky perfectly throughout is very difficult. There will be errors associated with it. In this project, it's tried to analyse the tracking accuracy of the 3.6 meter Telescope at Devasthal [5, 6] and find the RMS tracking errors for various exposure times. Data for analysis is taken from the telescope at different exposures covering different altitude and azimuth locations in the sky. In further chapters, detailed analysis is done.

Chapter 2

Instruments and Data

Instruments used for taking the image data from the telescope at Devasthal are;

2.1 AGU-camera

- Camera: Microline ML 402ME
- CCD pixel size: 9*9 μm^2
- CCD readnoise: 15 electrons
- CCD dark current: 0.5 e/s/pixel
- Field of view: 0.166 arcsec/pixel
- Waveband: 550 nm 750 nm
- CCD QE: 70% (550 nm), 80% (650 nm), 60% (750 nm)
- water cooled

2.2 Test camera

- Camera: Microline ML 402ME
- Field of view: 0.05768 arcsec/pixel
- air cooled
- other specs same as for AGU-Camera

2.3 Telescope Data

The following data is used for the analysis which was provided by the team at Devasthal. Observations are made from the telescope at Devasthal.

Main Port - Closed loop

1. Measurement 1
    - Alt [deg] : 63 to 50
    - Az [deg] : -56 to -59
    - Acquisition time : 30 s
2. Measurement 2
    - Alt [deg] : 79 to 75
    - Az [deg] : 20 to -35
    - Acquisition time : 30 s
3. Measurement 3
    - Alt [deg] : 30 to 41
    - Az [deg] : 124 to 138
    - Acquisition time : 30 s
4. Measurement 4 (10 arcmin Field of view radius)
    - Alt [deg] : 87 to 75
    - Az [deg] : 0 to -70
    - Acquisition time : 30 s

Side Port 1 - Closed loop

1. Measurement 1
    - Alt [deg] : 47 to 32
    - Az [deg] : -75 to -70
    - Acquisition time : 30 s

2. Measurement 2
    - Alt [deg] : 56 to 70
    - Az [deg] : 86 to 94
    - Acquisition time : 30 s
3. Measurement 3
    - Alt [deg] : 33 to 26
    - Az [deg] : -57 to -55
    - Acquisition time : 30 s
4. Measurement 4
    - Alt [deg] : 80.5 to 82
    - Az [deg] : 25 to -30
    - Acquisition time : 30 s

Side Port 2 - Closed loop

1. Measurement 1
    - Alt [deg] : 55 to 40
    - Az [deg] : -118 to -104
    - Acquisition time : 30 s
2. Measurement 2
    - Alt [deg] : 32 to 48
    - Az [deg] : 86 to 96
    - Acquisition time : 30 s
3. Measurement 3
    - Alt [deg] : 87 to 73
    - Az [deg] : 200 to 260
    - Acquisition time : 30 s

4. Measurement 4

    - Alt [deg] : 53 to 49
    - Az [deg] : -10 to -20
    - Acquisition time : 30 s

Chapter 3

Approach

The images taken of the objects in the sky through telescope are CCD images. Under each measurement, There are many frames captured with certain exposure time. Any star which is observed will have been shifted slightly from its position in the previous frame because of errors in tracking. Now, to find out the tracking errors in turn to find the tracking accuracy, centroid of the star in each of the frame is calculated.

The simplest approach in finding the centroid of a star in a frame is marginal sums approach or first moment distributions approach. In an image of size $2L+1 * 2L+1$ (where L is comparable to the size of star), intensity values of each pixel within the image are summed in both x and y directions starting from some rough pointer. The x, y center is computed as follows [7]

The marginal distributions of the star in the images are found from

$$I_i = \sum_{j=-L}^{j=L} I_{i,j} \tag{3.1}$$

and

$$J_j = \sum_{i=-L}^{i=L} J_{i,j} \tag{3.2}$$

where $I_{i,j}$ is the intensity (in ADU) at each x, y pixel; the mean intensities are determined from

$$\bar{I} = \frac{1}{2L+1} \sum_{i=-L}^{i=L} I_i \tag{3.3}$$

and

$$\overline{J} = \frac{1}{2L+1} \sum_{j=-L}^{j=L} J_j \tag{3.4}$$

and finally the intensity weighted centroid is determined using

$$x_c = \frac{\sum_{i=-L}^{i=L} (I_i - \overline{I}) x_i}{\sum_{i=-L}^{i=L} (I_i - \overline{I})} \tag{3.5}$$

for all $I_i - \overline{I} > 0$ and

$$y_c = \frac{\sum_{j=-L}^{j=L} (J_j - \overline{J}) y_i}{\sum_{j=-L}^{j=L} (J_j - \overline{J})} \tag{3.6}$$

for all $J_j - \overline{J} > 0$

For well sampled images, this method provides centroids as good as one fifth of a pixel.

After finding the centroids using the equations above for each of the measurement containing several frames, now we find moving averages over 30 sec over entire dataset. We convert these x, y centroid coordinates in terms of arcsec and then take the values with respect to zero-mean. To know the centroid position variation, over the entire data range, we plot the the moving averages calculated over 30 sec over time. This plot gives us information regarding shifts in the centroid in x and y directions frame by frame. Using the same centroid values for all the frames, we can find the RMS error for different time exposures.

1. For RMS error in 1 min, first we need to consider the moving average values of the frames comprising 1 min exposure.

2. Next, standard deviation is calculated in each x and y coordinates from the moving averages at each exposure of 1 min considered.

3. From the standard deviation values in x and y coordinates, RMS error is calculated as

$$RMS = \sqrt{x^2 + y^2} \tag{3.7}$$

4. These RMS errors in each interval of 1 min are calculated over entire span of observation data

5. The procedure is extended to RMS on 30 min and RMS on 50 min.

Thus, we get RMS errors for each of the exposures required.

Chapter 4

Results and Analysis

In this chapter, the results in each of the measurement is presented along with graphs.

4.1 Main Port - Closed loop

Measurement 1 : 20151130-200400-Tcam-MP-CL

1. Total number of frames: 1679

2. Exposure : 2 sec

3. Total time : 3358 sec (55.96 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

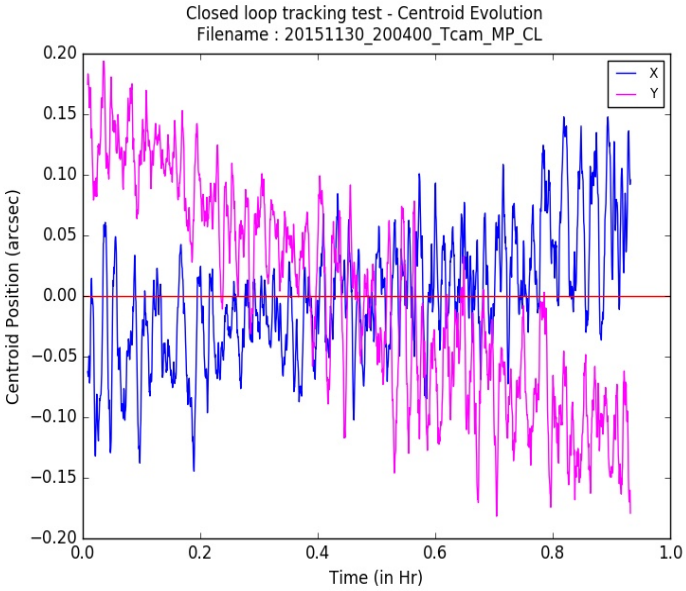


Fig. 4.1 Centroid evolution - 20151130-200400-Tcam-MP-CL

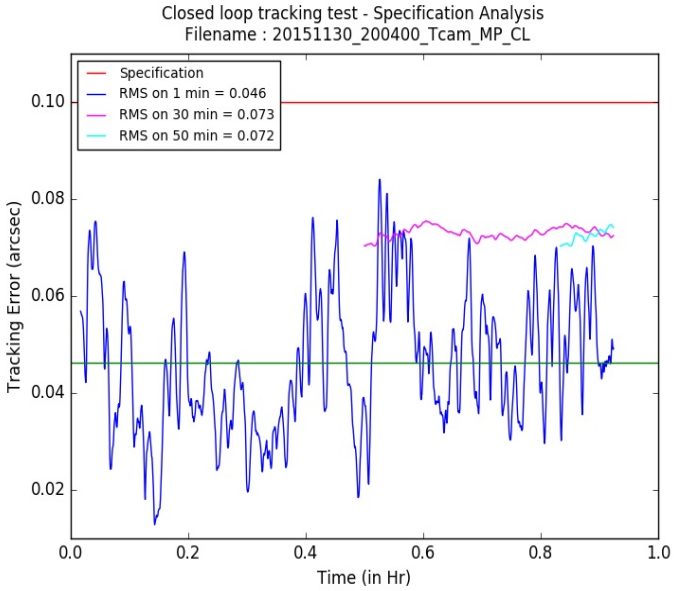


Fig. 4.2 Tracking error (RMS) - 20151130-200400-Tcam-MP-CL

- RMS error on 1 min : 0.046 arcsec
- RMS error on 30 min : 0.073 arcsec
- RMS error on 50 min : 0.072 arcsec

Measurement 2 : 20151130-212600-Tcam-MP-CL

1. Total number of frames: 4315

2. Exposure : 0.5 sec

3. Total time : 2157.5 sec (35.958 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

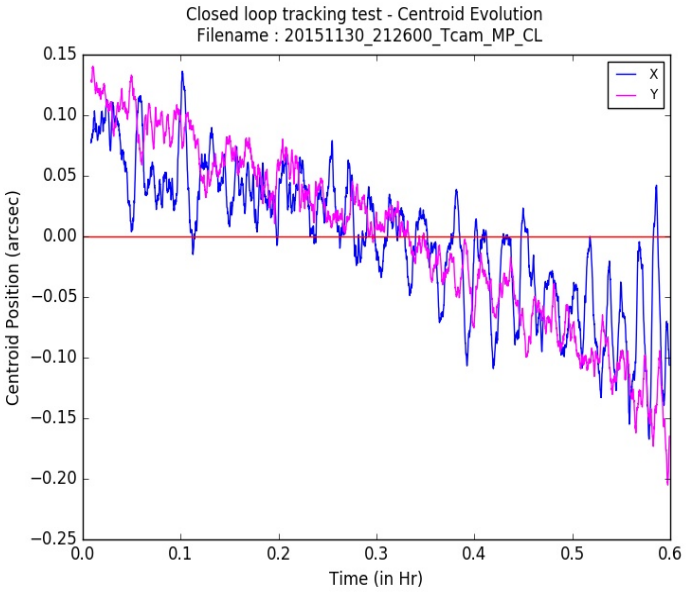


Fig. 4.3 Centroid evolution - 20151130-212600-Tcam-MP-CL

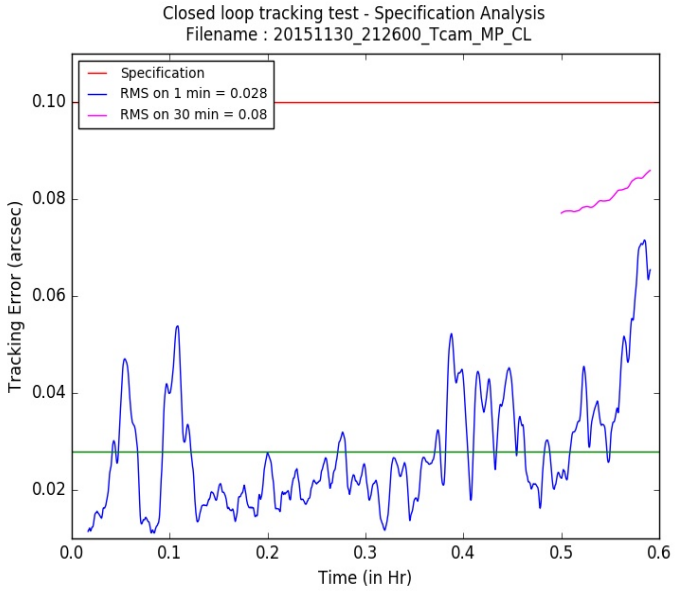


Fig. 4.4 Tracking error (RMS) - 20151130-212600-Tcam-MP-CL

- RMS error on 1 min : 0.028 arcsec
- RMS error on 30 min : 0.08 arcsec

Measurement 3 : 20151130-224500-Tcam-MP-CL

1. Total number of frames: 730
2. Exposure : 8 sec (frame 1-209), 4 sec (frame 210-730)
3. Total time : 3756 sec (62.6 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

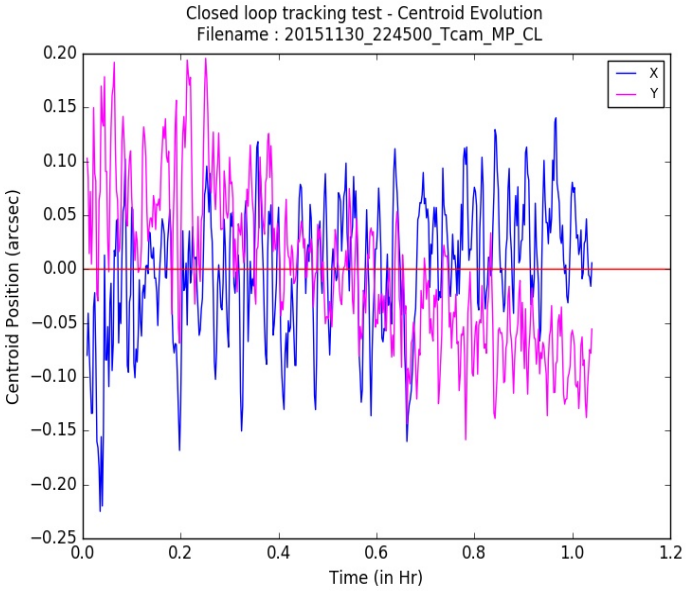


Fig. 4.5 Centroid evolution - 20151130-224500-Tcam-MP-CL

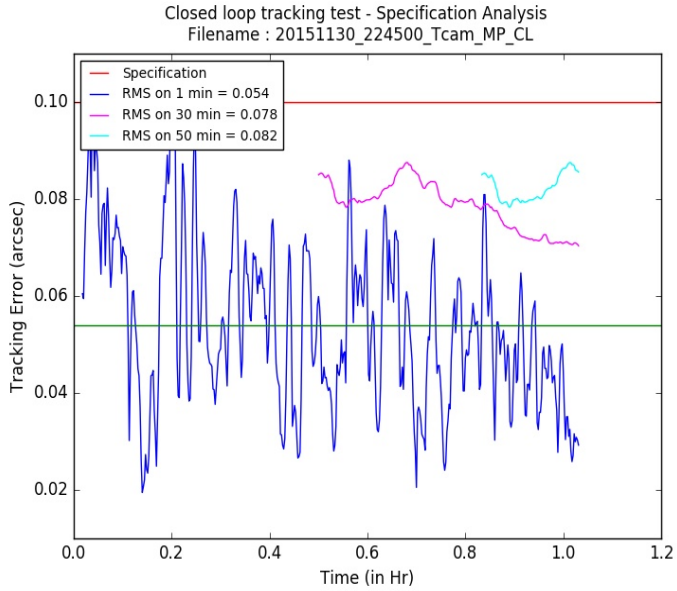


Fig. 4.6 Tracking error (RMS) - 20151130-224500-Tcam-MP-CL

- RMS error on 1 min : 0.054 arcsec
- RMS error on 30 min : 0.078 arcsec
- RMS error on 50 min : 0.082 arcsec

Measurement 4 : 20151202-234800-Tcam-MP-CL

1. Total number of frames: 627

2. Exposure : 6 sec

3. Total time : 3762 sec (62.7 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

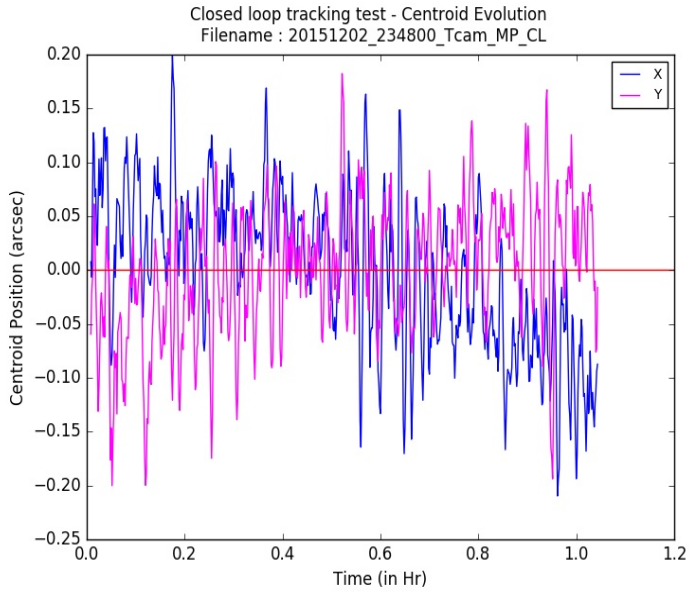


Fig. 4.7 Centroid evolution - 20151202-234800-Tcam-MP-CL

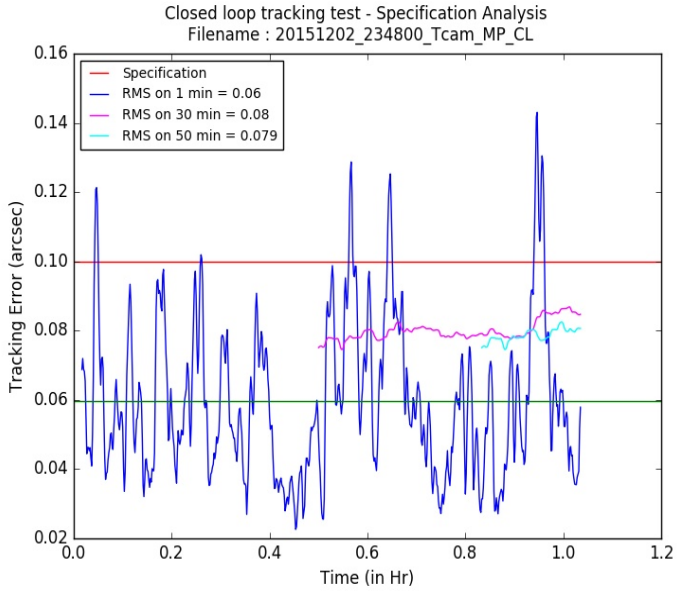


Fig. 4.8 Tracking error (RMS) - 20151202-234800-Tcam-MP-CL

- RMS error on 1 min : 0.06 arcsec
- RMS error on 30 min : 0.08 arcsec
- RMS error on 50 min : 0.079 arcsec

4.2 Side Port 1 - Closed loop

Measurement 1 : 20151107-031900-Tcam-SP1-CL

1. Total number of frames: 1163
2. Exposure : 3 sec
3. Total time : 3489 sec (58.15 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

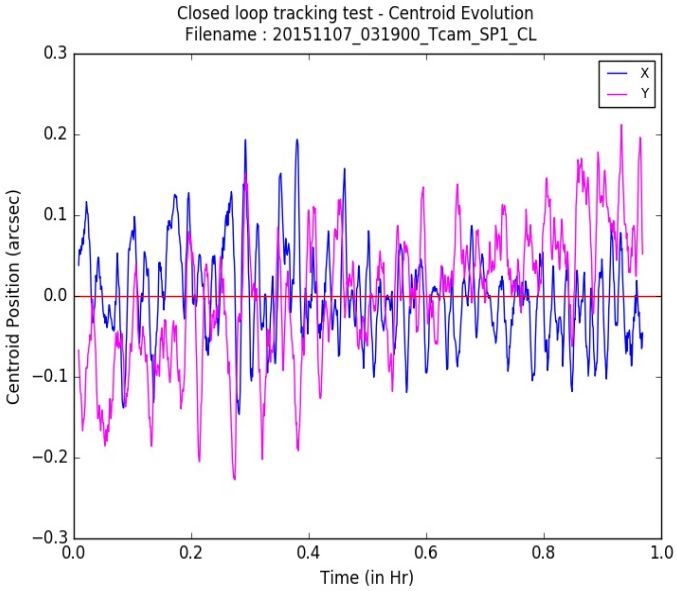


Fig. 4.9 Centroid evolution - 201511307-031900-Tcam-SP1-CL

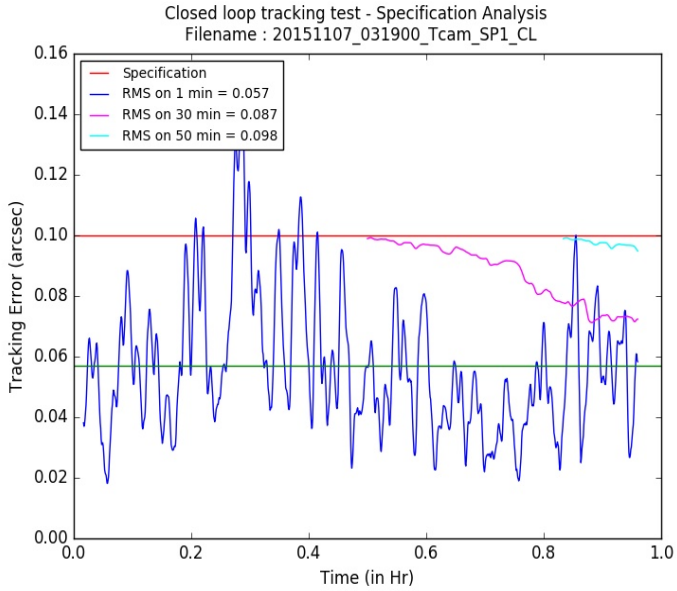


Fig. 4.10 Tracking error (RMS) - 20151107-031900-Tcam-SP1-CL

- RMS error on 1 min : 0.057 arcsec
- RMS error on 30 min : 0.087 arcsec
- RMS error on 50 min : 0.098 arcsec

Measurement 2 : 20151107-194900-Tcam-SP1-CL

1. Total number of frames: 1964

2. Exposure : 3 sec (frame 1-124), 1.5 sec (frame 125-1964)

3. Total time : 3132 sec (52.2 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

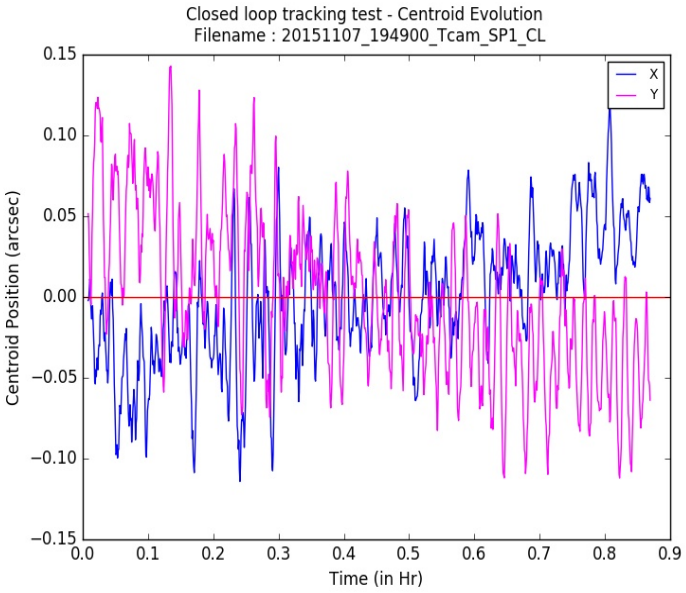


Fig. 4.11 Centroid evolution - 201511307-194900-Tcam-SP1-CL

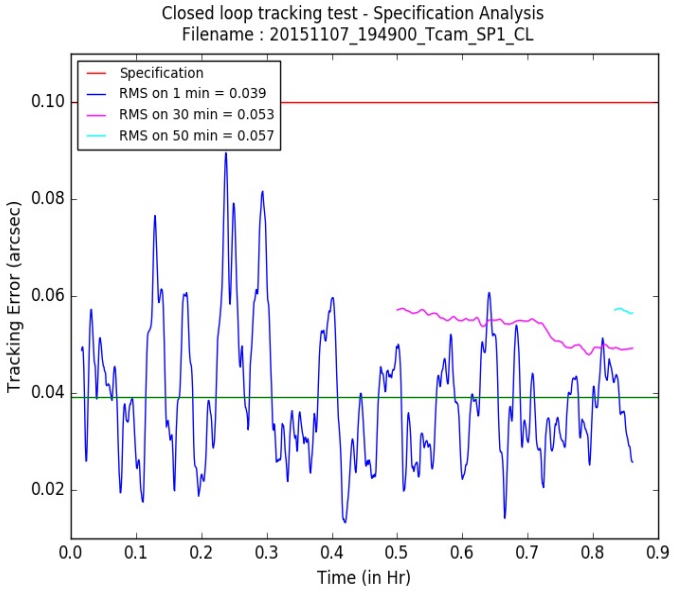


Fig. 4.12 Tracking error (RMS) - 20151107-194900-Tcam-SP1-CL

- RMS error on 1 min : 0.039 arcsec
- RMS error on 30 min : 0.053 arcsec
- RMS error on 50 min : 0.057 arcsec

Measurement 3 : 20151108-001400-Tcam-SP1-CL

1. Total number of frames: 851
2. Exposure : 4 sec
3. Total time : 3404 sec (56.73 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

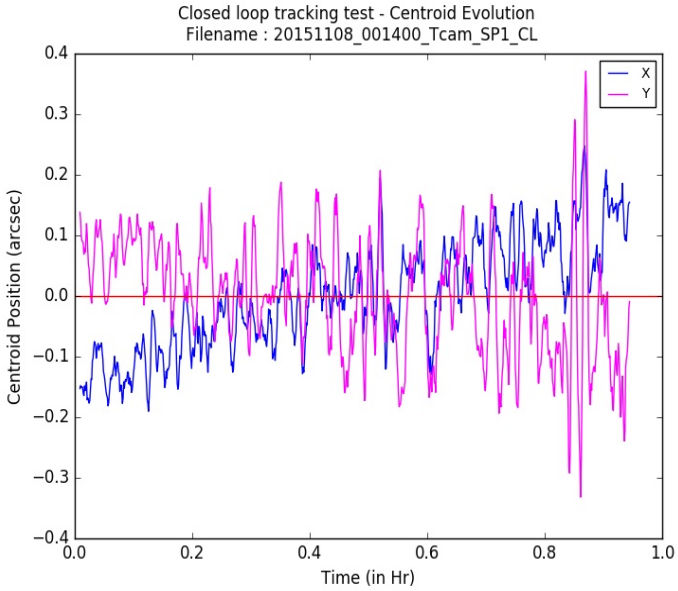


Fig. 4.13 Centroid evolution - 201511307-001400-Tcam-SP1-CL

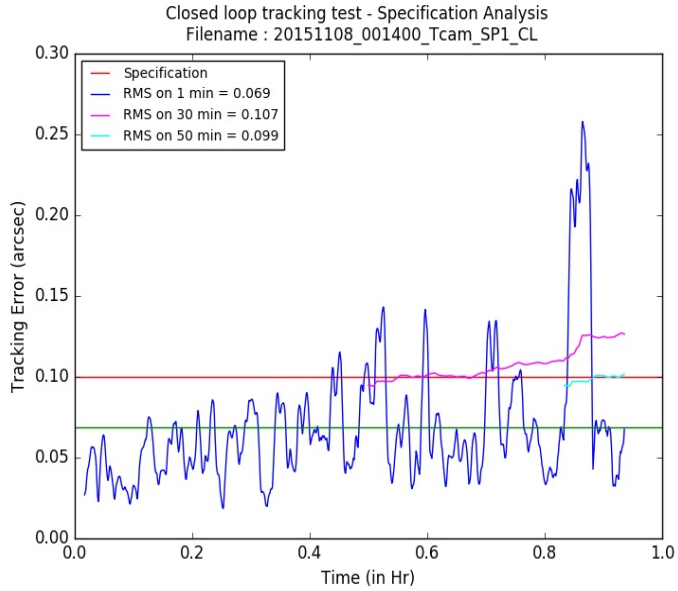


Fig. 4.14 Tracking error (RMS) - 20151107-001400-Tcam-SP1-CL

- RMS error on 1 min : 0.069 arcsec
- RMS error on 30 min : **0.0107 arcsec**
- RMS error on 50 min : 0.099 arcsec

Measurement 4 : 20151108-014000-Tcam-SP1-CL

1. Total number of frames: 1158

2. Exposure : 3 sec

3. Total time : 3474 sec (57.9 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

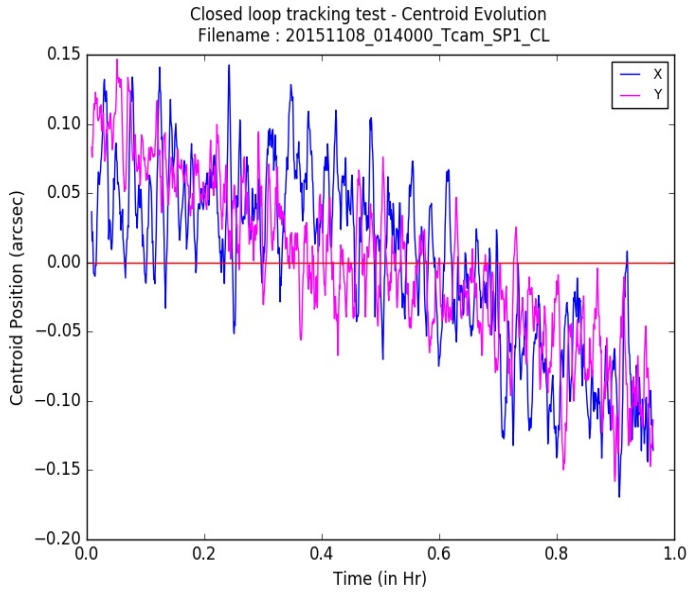


Fig. 4.15 Centroid evolution - 201511307-014000-Tcam-SP1-CL

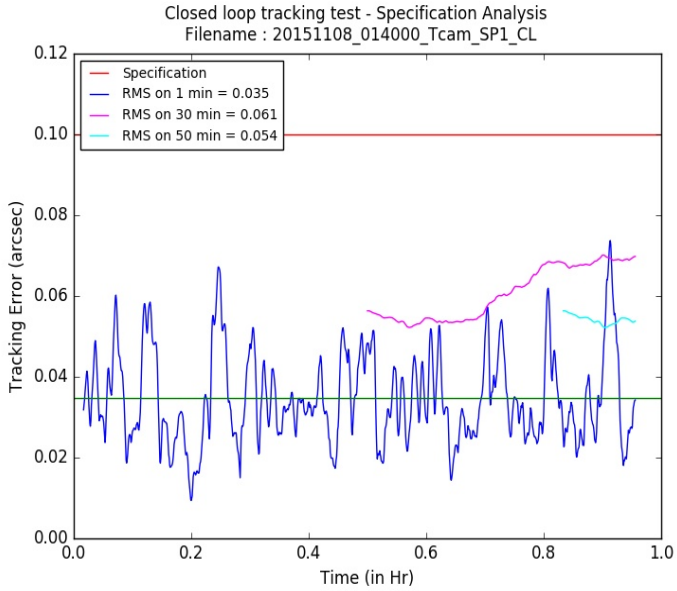


Fig. 4.16 Tracking error (RMS) - 20151107-014000-Tcam-SP1-CL

- RMS error on 1 min : 0.035 arcsec
- RMS error on 30 min : 0.061 arcsec
- RMS error on 50 min : 0.054 arcsec

4.3 Side Port 2 - Closed loop

Measurement 1 : 20151202-021500-Tcam-SP2-CL

1. Total number of frames: 386
2. Exposure : 10 sec
3. Total time : 3868 sec (64.33 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

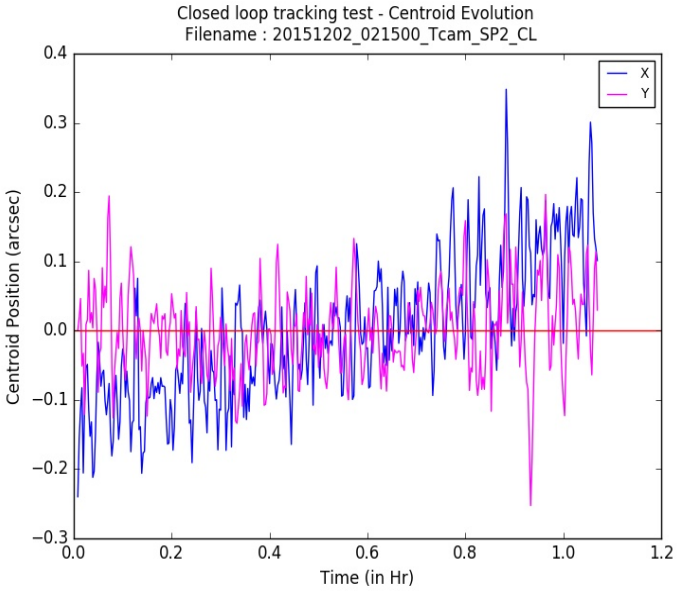


Fig. 4.17 Centroid evolution - 20151202-021500-Tcam-SP2-CL

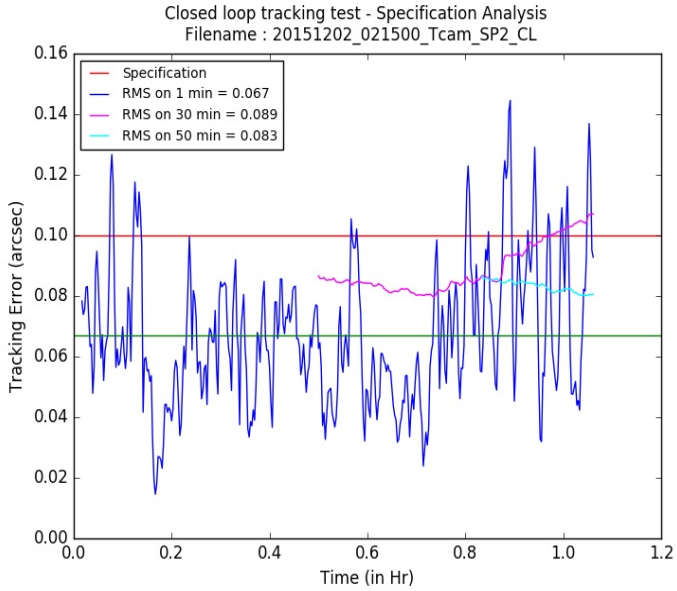


Fig. 4.18 Tracking error (RMS) - 20151202-021500-Tcam-SP2-CL

- RMS error on 1 min : 0.067 arcsec
- RMS error on 30 min : 0.089 arcsec
- RMS error on 50 min : 0.083 arcsec

Measurement 2 : 20151202-184900-Tcam-SP2-CL

1. Total number of frames: 706

2. Exposure : 5 sec

3. Total time : 3530 sec (58.83 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

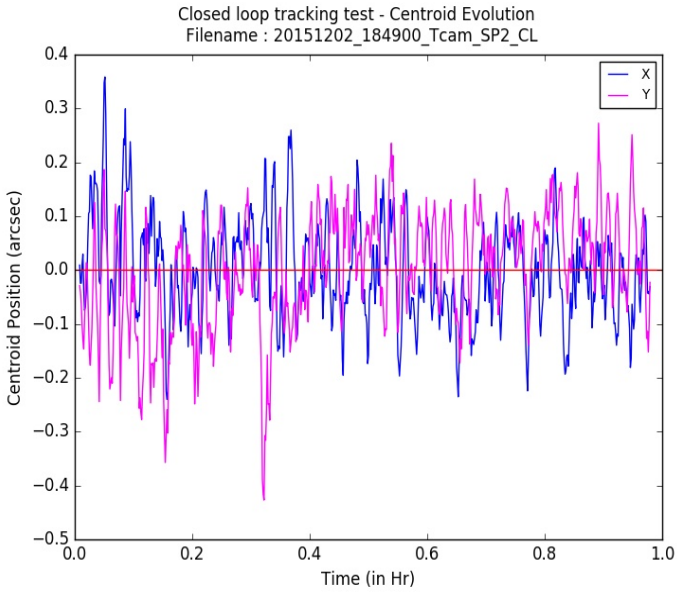


Fig. 4.19 Centroid evolution - 20151202-184900-Tcam-SP2-CL

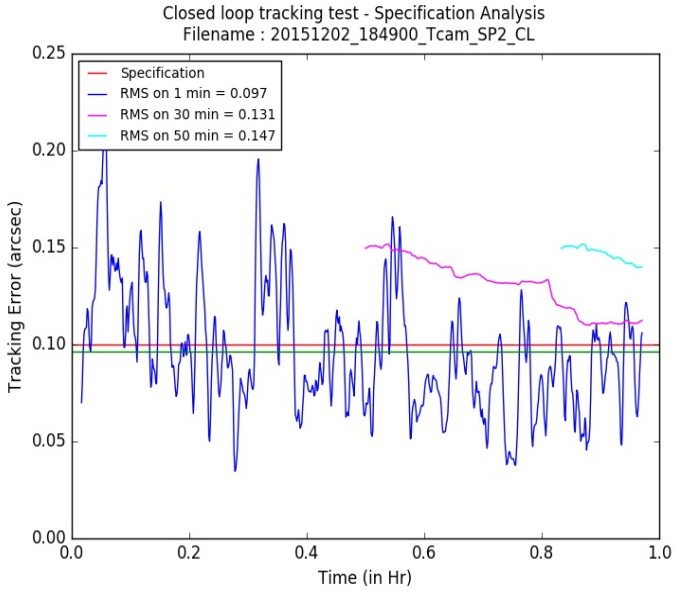


Fig. 4.20 Tracking error (RMS) - 20151202-184900-Tcam-SP2-CL

- RMS error on 1 min : 0.097 arcsec
- RMS error on 30 min : **0.131 arcsec**
- RMS error on 50 min : **0.147 arcsec**

Measurement 3 : 20151202-205000-Tcam-SP2-CL

1. Total number of frames: 393
2. Exposure : 10 sec
3. Total time : 3930 sec (65.5 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

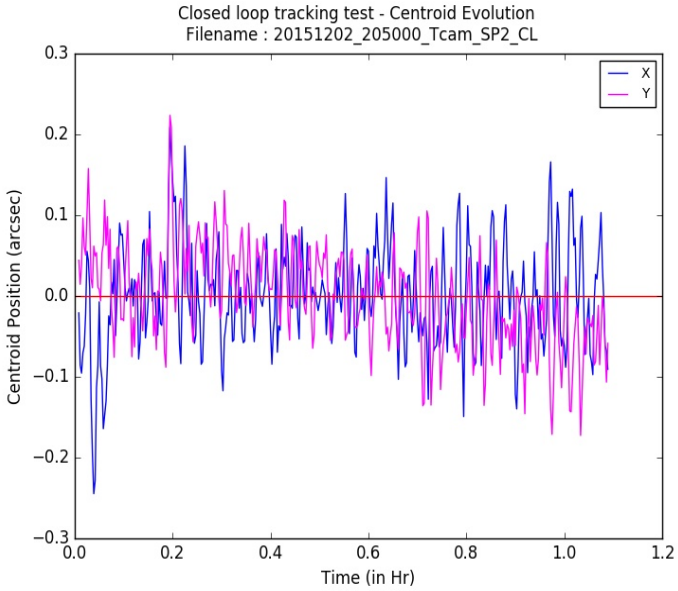


Fig. 4.21 Centroid evolution - 20151202-205000-Tcam-SP2-CL

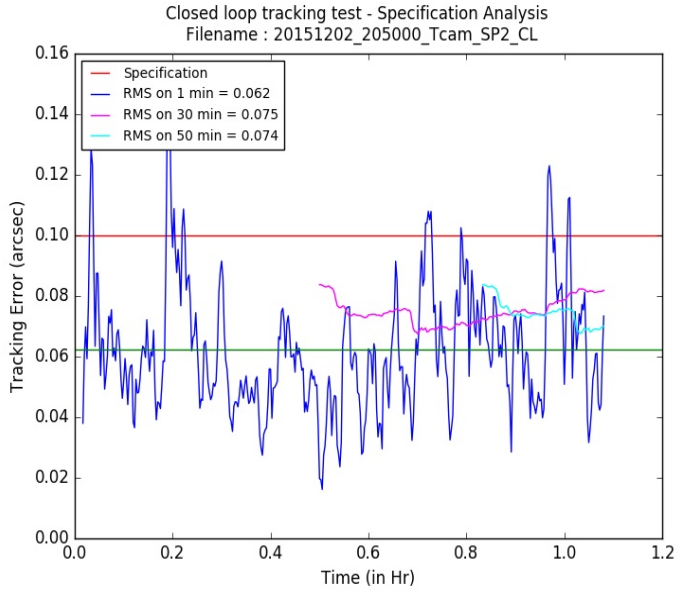


Fig. 4.22 Tracking error (RMS) - 20151202-205000-Tcam-SP2-CL

- RMS error on 1 min : 0.062 arcsec
- RMS error on 30 min : 0.075 arcsec
- RMS error on 50 min : 0.074 arcsec

Measurement 4 : 20151202-220600-Tcam-SP2-CL

1. Total number of frames: 879

2. Exposure : 4 sec

3. Total time : 3516 sec (58.6 min)

In the following figures, we can see the variations in centroid coordinates and RMS errors over time

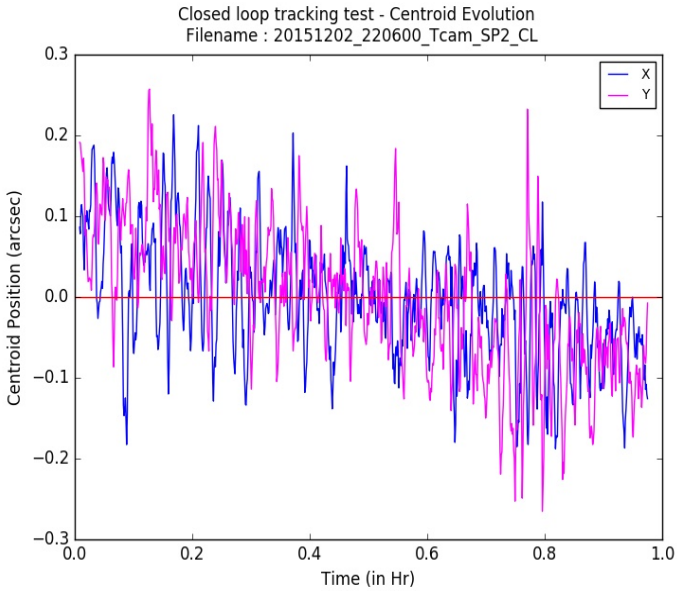


Fig. 4.23 Centroid evolution - 20151202-220600-Tcam-SP2-CL

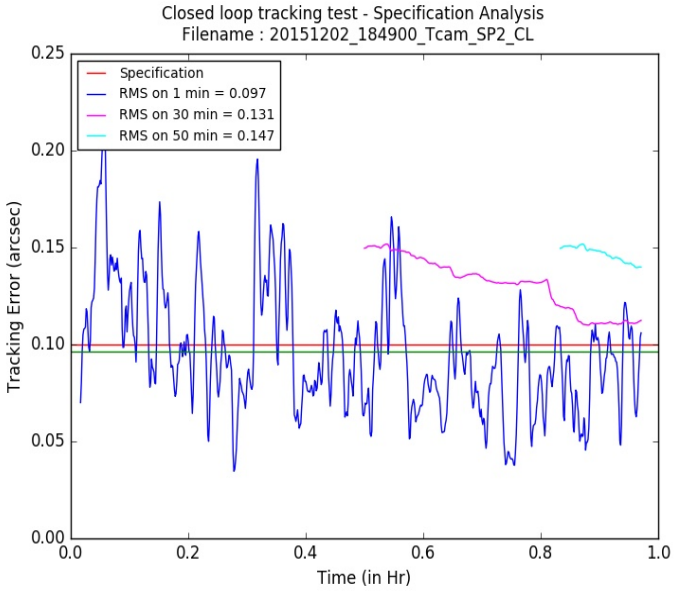


Fig. 4.24 Tracking error (RMS) - 20151202-220600-Tcam-SP2-CL

- RMS error on 1 min : 0.071 arcsec
- RMS error on 30 min : 0.096 arcsec
- RMS error on 50 min : 0.097 arcsec

Chapter 5

Conclusions

1. This analysis was meant for the telescope adjustments to get the tracking and image quality specifications. The performance have been tested on Main Port, Side Port 1 and Side Port 2.

2. Training related to Telescope control system, rotator, adapter and guider were not undertaken.

3. A total of 12 tracking measurements have been taken altogether on all three ports, out of which there are three values 0.107, 0.131, 0.147 which are greater than the specified value of 0.1 arcsec.

4. From the measurements, it can be seen that the tracking of the telescope is well within the specified values of 0.1 arcsec RMS for less than 1 min and 0.11 arcsec RMS for less than 1 hour.

Part II

Temperature map and optical depth of star forming molecular clouds from Planck satellite data

Chapter 6

Introduction

6.1 What are molecular clouds?

A molecular cloud[8] is an accumulation of interstellar gas and dust. The temperature of these clouds vary from 10 to 30 kelvin. Primarily, these consist of molecular hydrogen along with traces of CO and other species. Their size can be as big as 600 light years and mass as high as several million solar masses. The density of these regions is also high ranging in the order of $10^{12} particles/m^3$. Particles in these regions are mostly in molecular form rather than atomic, and thus these high dense regions are called molecular clouds. These molecular clouds contain enough mass to make a million stars like sun.

6.2 Why do we study them?

Molecular clouds play an important galactic role in the formation of massive stars. These massive stars, formed in them, provide the main energy sources for the interstellar medium, partly by destroying their birth clouds and recycling their matter back into more diffuse forms. Thus an understanding of the life cycle of molecular clouds is of central importance not only for understanding how stars form, but also for understanding the dynamics of the interstellar medium and ultimately the evolution of galaxies as a whole. Hundreds of different types of molecules have been detected in the molecular clouds such as water, ammonia, ethyl alcohol, even sugar and amino acids like glycine ($C_2H_5NO_2$) the basic molecules of life. An amino acid is an essential building block for proteins, which is foundation for DNA, which in turn defines life. This shows that other life forms may exist out there other than earth. Molecular clouds present a wide range of physical conditions (illumination, density,

star-forming activity), and therefore they are ideal for studying the emission properties of dust grains and their evolution.

6.3 Understanding the molecular clouds

We see that most of the mass of a molecular cloud is molecular hydrogen. Even then it's invisible under quiescent interstellar conditions. Because of this, we go for observations of emission and absorption from dust and rotational lines from species such as CO and its isotopologues. These help in determining the properties of molecular clouds. It has been found in the recent studies that dense cores ($n > 10^4 cm^{-3}$) exist preferentially in the small percentage of molecular clouds with high column density. In order to understand why these dense cores form only in certain regions, how long these cores can survive without collapsing or dispersing, and a few other star formation issues, we need to determine basic properties of molecular clouds such as the temperature, optical depth, density distributions, spectral emissivity index and internal motions of the gas and dust[9].

6.4 Planck Satellite

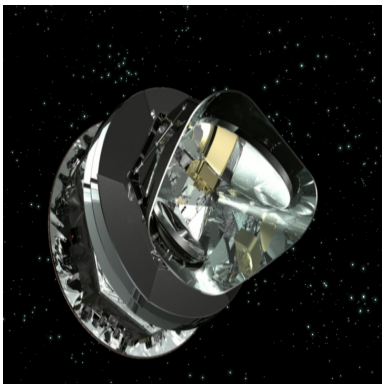
Planck [10, 11] is a European Space Agency mission designed to image the temperature and polarization anisotropies of the Cosmic Background Radiation Field over the whole sky, with unprecedented sensitivity and angular resolution at microwave and infra-red frequencies. Built at the Cannes Mandelieu Space Center by Thales Alenia Space, and created as a medium-sized mission for ESA's Horizon 2000 long-term scientific programme, *Planck* was launched in May 2009, reaching the Earth/Sun L2 point by July, and by February 2010 had successfully started a second all-sky survey. On 21 March 2013, the mission's first all-sky map of the cosmic microwave background was released, with an expanded release including polarization data in February 2015.

The mission had a wide variety of scientific aims, including:

1. high resolution detections of both the total intensity and polarization of primordial CMB anisotropies

2. creation of a catalogue of galaxy clusters through the Sunyaev–Zel'dovich effect

3. observations of the gravitational lensing of the CMB, as well as the integrated Sachs–Wolfe effect



(a) [12]

(b) [13]

Fig. 6.1 Planck Satellite

Table 6.1 Low frequency instrument[3] parameters of Planck satellite

Frequency (GHz)	Bandwidth ($\Delta\nu/\nu$)	Resolution (arcmin)	Sensitivity(total intensity), $\Delta T/T, 10^{-6}$	Sensitivity (Polarization), $\Delta T/T, 10^{-6}$
30	0.2	33	2.0	2.8
44	0.2	24	2.7	3.9
70	0.2	14	4.7	6.7

4. observations of bright extragalactic radio (active galactic nuclei) and infrared (dusty galaxy) sources

5. observations of the Milky Way, including the interstellar medium, distributed synchrotron emission and measurements of the Galactic magnetic field, and

6. studies of the Solar System, including planets, asteroids, comets and the zodiacal light

The spacecraft carries two instruments: the Low Frequency Instrument (LFI) [reference] and the High Frequency Instrument (HFI) [reference]. Both instruments can detect both the total intensity and polarization of photons, and together cover a frequency range of nearly 830 GHz (from 30 to 857 GHz). The cosmic microwave background spectrum peaks at a frequency of 160.2 GHz.

Table 6.2 High frequency instrument[4] parameters of Planck satellite

Frequency (GHz)	Bandwidth ($\Delta\nu/\nu$)	Resolution (arcmin)	Sensitivity(total intensity), $\Delta T/T, 10^{-6}$	Sensitivity (Polarization), $\Delta T/T, 10^{-6}$
100	0.33	10	2.5	4.0
143	0.33	7.1	2.2	4.2
217	0.33	5.5	4.8	9.8
353	0.33	5.0	14.7	29.8
545	0.33	5.0	14.7	N/A
857	0.33	5.0	6700	N/A

6.5 IRAS

The Infrared Astronomical Satellite (IRAS)[14, 15] was the first-ever space-based observatory to perform a survey of the entire sky at infrared wavelengths. The telescope was a joint mission of the United States (NASA), the United Kingdom (SERC) and the Netherlands (NIVR). Infrared sources were observed in 12, 25, 60 and 100 μm wavelengths, with resolutions ranging from 30 arcsec at 12 μm to 2 arcmin at 100 μm. IRAS was designed to catalogue fixed sources, so it scanned the same region of sky several times.

Fig. 6.2 IRAS satellite[1]

6.6 Scope

This project aims at creating a general tool to generate temperature plot, optical depth map, spectral emissivity index map and also the correlations of these parameters of any given molecular cloud. Given the data of the molecular region in 6 bands of Planck HFI spectral data and also iris 100μm data from IRAS, we fit a modified blackbody curve to get the

temperature at each pixel of the image. This will enable further analysis and studies of molecular clouds.

In chapter 2, we discuss about the data used for the reduction process to generate temperature maps. Image processing methods are used to have uniform data types with the help of PSF and Kernel generation and then convolving the original images with kernels.

In chapter 3, we discuss about the emission spectrum, background correction and the approach used for fitting the modified blackbody curve. Later, we compare the fitting for a location in both molecular and non-molecular region to see the differences and efficiency of fitting.

In chapter 4, we validate our results of Taurus molecular cloud with already availabele results and further extend the analysis to two more molecular clouds Lambda Ori and Perseus.

Further in chapter 5, we conclude with the summary of the results obtained and shortcomings of methods, further possible refinements to achieve better results and use of these results in further analysis by other astronomers and astrophysicists.

Chapter 7

Observations

7.1 Data

HFI data of Planck, along with iris $100\mu m$ data from IRAS is used for the analysis and generating the tool for mapping various parameters. The bands used are 100, 143, 217, 353, 545, 857 GHz from HFI data and $100\mu m$ band is used from IRAS. The data are taken from Skyview[16] with following parameters.

Planck data, 100 GHz:
Regime : Millimeter
Pixel scale : 1.7' (arcmin)
Pixel units : K (brightness temperature)
Resolution : 10'
Coordinate system : Galactic
Projection : Tan
143 GHz:
Regime : Millimeter
Pixel scale : 1.7' (arcmin)
Pixel units : K (brightness temperature)
Resolution : 7.1'
Coordinate system : Galactic
Projection : Tan
217 GHz:
Regime : Millimeter
Pixel scale : 1.7' (arcmin)
Pixel units : K (brightness temperature)
Resolution : 5.5'

Coordinate system : Galactic

Projection : Tan

353 GHz:

Regime : Millimeter

Pixel scale : 1.7’ (arcmin)

Pixel units : K (brightness temperature)

Resolution : 5.0’

Coordinate system : Galactic

Projection : Tan

545 GHz:

Regime : Millimeter

Pixel scale : 1.7’ (arcmin)

Pixel units : MJy/Sr (Spectral Flux density)

Resolution : 5.0’

Coordinate system : Galactic

Projection : Tan

857 GHz:

Regime : Millimeter

Pixel scale : 1.7’ (arcmin)

Pixel units : MJy/Sr (Spectral Flux density)

Resolution : 5.0’

Coordinate system : Galactic

Projection : Tan

IRAS, iris 100:

Regime : Infrared

Frequency: 3000 GHz, Bandpass: 2.5-3.6 THz

Pixel scale : 1.5’ (arcmin)

Pixel units : MJy/Sr (Spectral Flux density)

Resolution : 2’

Coordinate system : Galactic

Projection : Tan

7.2 Data for Validation

For validating the results found from the method followed, Taurus molecular cloud is used as the reference molecular cloud for which results are available [25]. Data for this cloud is taken in the bands mentioned in 7.1 with following specifications.

Name: Taurus Molecular cloud

Reference longitude: 173.0°

Reference latitude: -16.0°

Dimension of the image: 700*700

Image size (degrees): 19.833

Pixel scale: 1.7' (including iris 100 image)

For all bands, we obtain images with size 19.833 °*19.833 ° window centered at l = 173.0 ° and b = -16.0 °

7.2.1 Unit Conversion

The images of different band considered here have different pixel units. 100, 143, 217, 353 GHz images have brightness temperature as their pixel units where as 545, 857 GHz and iris 100 (3000GHz) images have spectral flux density as their pixel units. It's necessary to find a conversion parameter between these images to have a uniform analysis[17].

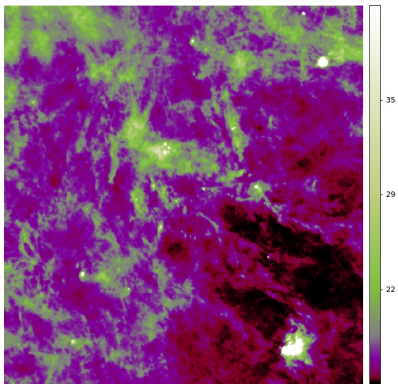
7.2.2 Spectral Flux Density/ Spectral Brightness

Spectral flux density/ Specific intensity[17] is the quantity that describes the rate at which energy is transferred by electromagnetic radiation through a real or virtual surface, per unit surface area and per unit wavelength per solid angle. It's SI unit is $Wm^{-2}Hz^{-1}Sr^{-1}$ or Jansky/Sr in non-SI. The quantitative definition is given as:

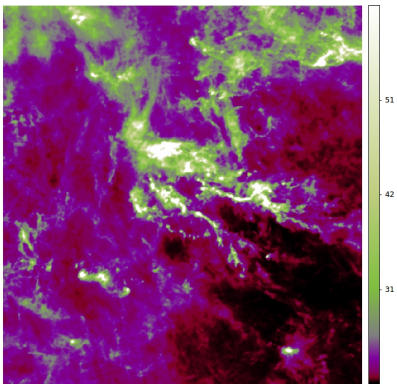
$$I_\nu = \frac{dP}{cos\theta.d\sigma.d\nu.d} \tag{7.1}$$

7.2.3 Brightness Temperature

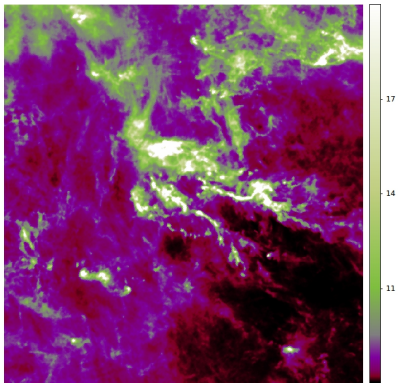
Brightness temperature[17] is the temperature a black body in thermal equilibrium with its surroundings would have to be to duplicate the observed intensity of a grey body object at a frequency ν. This concept is extensively used in radio astronomy and planetary science. It is not a temperature as usually understood. It characterizes radiation, and depending on the mechanism of radiation can differ considerably from the physical temperature of a radiating



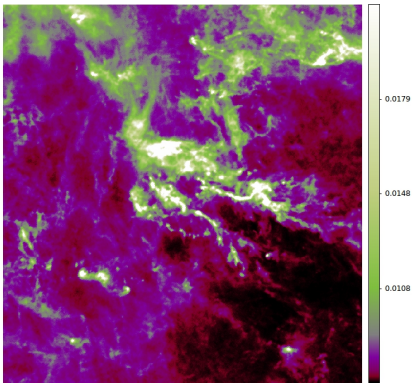
(a) 3000 GHz



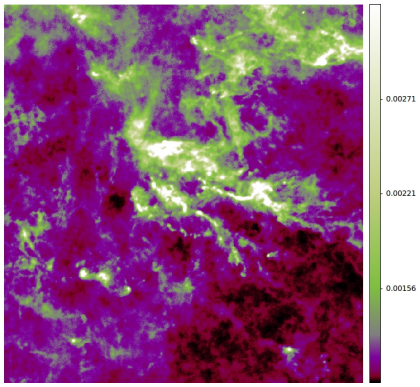
(b) 857 GHz



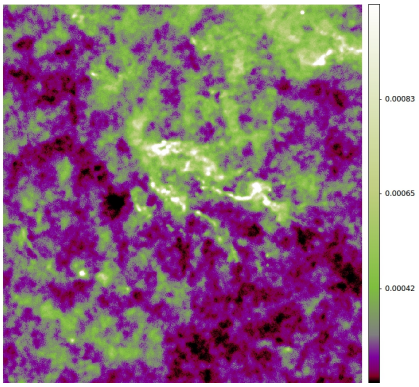
(c) 545 GHz



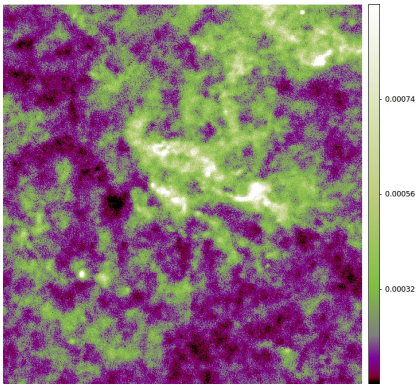
(d) 353 GHz



(e) 217 GHz



(f) 143 GHz



(g) 100 GHz

Fig. 7.1 IRAS and HFI maps of Taurus molecular cloud, 100, 143, 217, 353 GHz in K, 545, 857, 3000 GHz in MJy/Sr (without conversion)

body. For a black body, *Planck's* law gives:

$$I_\nu = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT}}-1} \tag{7.2}$$

Where, I_ν is the amount of energy emitted per unit surface area per unit time per unit solid angle and in the frequency range between ν and $\nu + d\nu$; T is the temperature of the black body; h is Planck's constant; ν is frequency; c is the speed of light; and k is Boltzmann's constant.

For a grey body, the spectral radiance is a portion of the black body radiance, determined by the emissivity ε, which means reciprocal of brightness temperature is,

$$T_b^{-1} = \frac{k}{h\nu} ln\left[1 + \frac{e^{\frac{h\nu}{kT}}-1}{\varepsilon}\right] \tag{7.3}$$

At low frequencies when $h\nu << kT$, we can use Rayleigh-Jeans law:

$$I_\nu = \frac{2\nu^2 kT}{c^2} \tag{7.4}$$

So that brightness temperature can simply be written as:

$$T_b = \varepsilon T \tag{7.5}$$

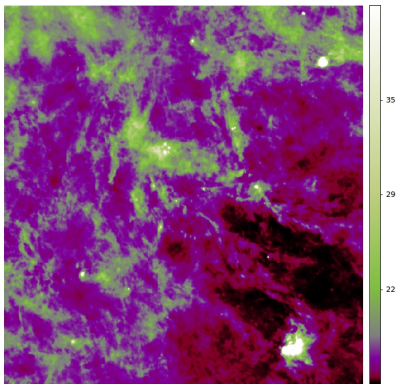
In general, the brightness temperature is a function of ν, and only in the case of blackbody radiation it's the same at all frequencies. The brightness temperature can be used to calculate the spectral index of a body, in the case of non-thermal radiation.

Conversion : The conversion between the spectral brightness and brightness temperature is given by the relation[18]

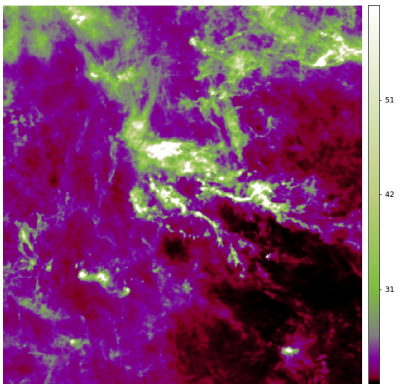
$$dT_b = \frac{c^2}{2\nu^2 k} dI_\nu [Kb] \tag{7.6}$$

7.3 Problems of Resolution

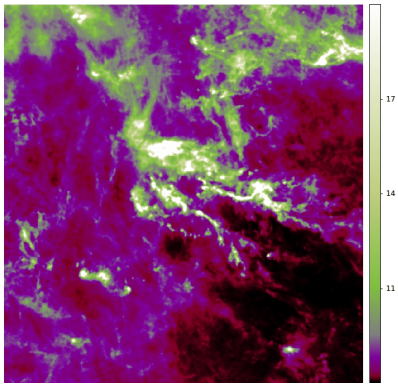
The data we considered for the temperature from modified blackbody curve fit has different angular resolution over different bands. Therefore they cannot be used for fitting the data from pixels to blackbody curve. It is needed to convolve the data images with their respective kernels to get images with same angular resolution, considering an image as the reference image, which is of lowest angular resolution[19].



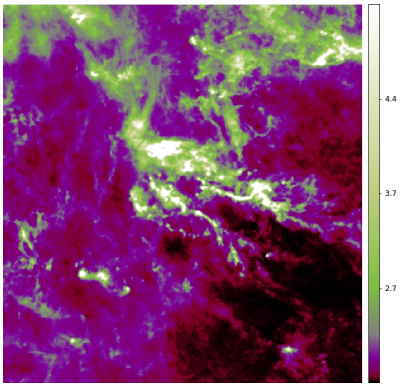
(a) 3000 GHz



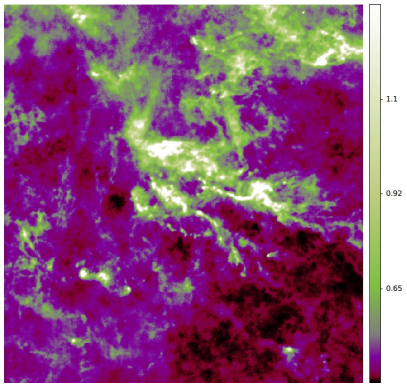
(b) 857 GHz



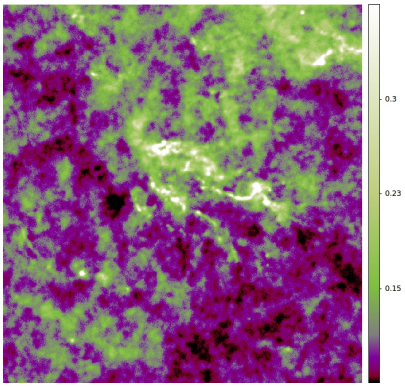
(c) 545 GHz



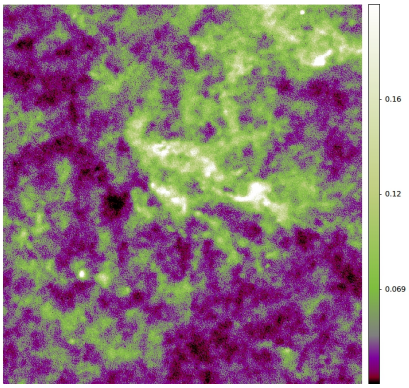
(d) 353 GHz



(e) 217 GHz



(f) 143 GHz



(g) 100 GHz

Fig. 7.2 IRAS and HFI maps of Taurus molecular cloud, in MJy/Sr (with conversion)

For the data considered in this project, angular resolution are as follows:
100 GHz : 10'
143 GHz : 7.1'
217 GHz : 5.5'
353 GHz : 5.0'
545 GHz : 5.0'
857 GHz : 5.0'
3000 GHz (iris 100) : 2.0'

100 GHz data is not considered for the modified blackbody curve fit as the resolution of the images is too bad and would affect other images' data if they are blurred to the resolution of 10'. Therefore, 143 GHz image is considered as the reference image and all the other images are blurred to the resolution of 143 GHz using convolution. To undertake this process, respective kernels are required for each image based on the point spread function.

7.3.1 Point Spread Function

The point spread function (PSF) describes the response of an imaging system to a point source or point object. A more general term for the PSF is a system's impulse response, the PSF being the impulse response of a focused optical system. The ideal point spread function (PSF) is the three-dimensional diffraction pattern of light emitted from an infinitely small point source in the specimen and transmitted to the image plane through a high numerical aperture (NA) objective. It is considered to be the fundamental unit of an image in theoretical models of image formation. When light is emitted from such a point object, a fraction of it is collected by the objective and focused at a corresponding point in the image plane. However, the objective lens does not focus the emitted light to an infinitely small point in the image plane. Rather, light waves converge and interfere at the focal point to produce a diffraction pattern of concentric rings of light surrounding a central, bright disk, when viewed in the x-y plane. The radius of disk is determined by the NA, thus the resolving power of an objective lens can be evaluated by measuring the size of the Airy disk. The image of the diffraction pattern can be represented as an intensity distribution as shown in Figure 7.3. The bright central portion of the Airy disk and concentric rings of light correspond to intensity peaks in the distribution. In Figure 7.3, relative intensity is plotted as a function of spatial position for PSFs from objectives having numerical apertures of 0.3 and 1.3. The full-width at half maximum (FWHM) is indicated for the lower NA objective along with the Rayleigh limit.

When we study spectral energy distribution of astronomical objects, we need to combine observations from camera with very different PSFs[20], with varying FWHM (full width half maximum) to achieve a wide range of wavelengths. Comparing Images with structures on

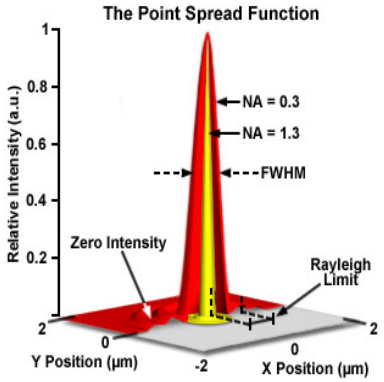


Fig. 7.3 Point Spread function[2]

multiple/spatial scales obtained with different PSFs directly result in unphysical intensity ratios. Intensity ratios must be calculated from images with a common PSF. Therefore, it is required to find convolution kernels that transforms the image taken with different instruments into a common PSF, so as to create image cubes in which each pixel corresponds to the same sky region for all cameras used.

The PSF $\psi_j(x,y,x',y')$ of a camera j gives the measured intensity at (x,y) produced by a point source with unit flux at the point (x',y'), where we use the cartesian coordinates (x,y) to denote positions in a small region of the sky[21]. With this definition, PSF has normalization:

$$\iint \psi_j(x,y,x',y')\,dx\,dy = 1 \tag{7.7}$$

for any source position (x',y').

It is often possible to approximate the PSF (denoted by ψ from now on) as constant across the useful field of view of the camera, so $\psi_j(x,y,x',y') = \psi_j(x-x',y-y')$. The observed image $I_j(x,y)$will then be the convolution of source $S(x,y)$ with the PSF ψ:

$$I_j(x,y) = \iint S(x',y')\,\psi_j(x-x',y-y')\,dx\,dy = (S \star \psi_j)(x,y) \tag{7.8}$$

7.3.2 Kernels

Given two cameras A and B, with PSF ψ_A and ψ_B , the images will be different for both cameras, even if the spectral responses are similar. A convolution kernel helps in transforming the image observed by one camera into an image corresponding to PSF of another camera[21].

The convolution kernel $K\{A \Rightarrow B\}$ from A to B should satisfy:

$$I_B(x,y) = \iint I_A(x',y')K\{A \Rightarrow B\}(x-x',y-y')\,dx'\,dy' = (I_A \star K\{A \Rightarrow B\})(x,y) \tag{7.9}$$

where I_A and I_B are the observed images by the cameras A and B respectively.

Given two cameras A and B, with PSF ψ_A and ψ_B, we see $K\{A \Rightarrow B\}$ that fulfills above equation. Thus,

$$(S \star \psi_B) = I_B = (I_A \star K\{A \Rightarrow B\}) = (S \star \psi_A \star K\{A \Rightarrow B\}) \tag{7.10}$$

For any astronomical source S, so the convolution kernel must satisfy

$$\psi_B = (\psi_A \star K\{A \Rightarrow B\}) \tag{7.11}$$

With the normalization condition, kernel must have

$$\iint K\{A \Rightarrow B\}(x,y)\,dx\,dy = 1 \tag{7.12}$$

Taking the 2-dimensional Fourier transform of equation 7.11, we obtain

$$FT(\psi_B) = FT(\psi_A \star K\{A \Rightarrow B\}) = FT(\psi_A)FT(K\{A \Rightarrow B\}) \tag{7.13}$$

This can be inverted to obtain

$$K\{A \Rightarrow B\} = FT^{-1}\left(FT(\psi_B)\frac{1}{FT(\psi_A)}\right) \tag{7.14}$$

where FT and FT^{-1} stand for Forier transform and its inverse Fourier transform respectively.

To avoid high frequency components of the kernel, we introduce a filter f_A in the kernel construction

$$K\{A \Rightarrow B\} = FT^{-1}\left(FT(\psi_B)\frac{f_A}{FT(\psi_A)}\right) \tag{7.15}$$

where f_A is a suitable low-pass filter

7.4 PSF Generation

In this case, we consider the PSF to be a gaussian[21]. So we generate a gaussian beam and then for each PSF profile, we generate kernels based on the FWHM values. Gaussian PSFs

are of the form,

$$\psi(\theta) = \frac{1}{2\pi\sigma^2} exp\left(\frac{-\theta^2}{2\sigma^2}\right) \tag{7.16}$$

where $FWHM = 2\sigma\sqrt{2ln2}$.

FWHM

From the parameters of the data, it's known that resolution of the image is the FWHM for the PSF of the that image. The resolution of the image is given in arcmin. Now, FWHM is to be calulated in terms of number of pixels, which means resolution of each image has to be divided by its pixel scale. Then, using the equation

$$\sigma = FWHM\frac{1}{2\sqrt{2ln2}}$$

Standard deviation can be calculated and hence the PSF for each of the image

Table 7.1 FWHM and standerd deviation of the images(in terms of pixels)

Frequency (GHz)	Resolution(arcmin)	Pixel Scale(arcmin/pixel)	FWHM (pixels)	S.D σ
143	7.1	1.7	4.1764	1.7735
217	5.5	1.7	3.2352	1.3738
353	5.0	1.7	2.9411	1.2489
545	5.0	1.7	2.9411	1.2489
857	5.0	1.7	2.9411	1.2489
3000	2.0	1.7	1.1764	0.4995

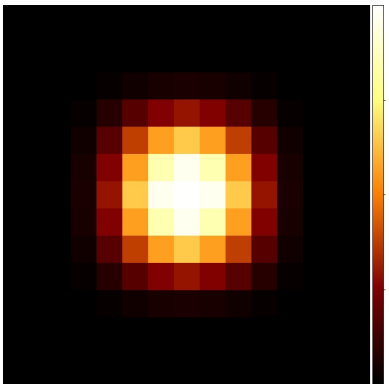
Using the standard deviation values in Table 7.1, PSFs are generated.

7.5 Kernel Generation

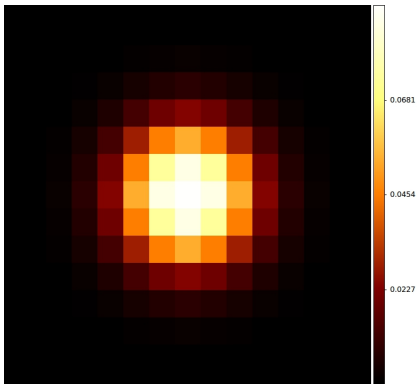
Generation of kernel involves following procedure[22]:

Input the PSF and correct for missing data

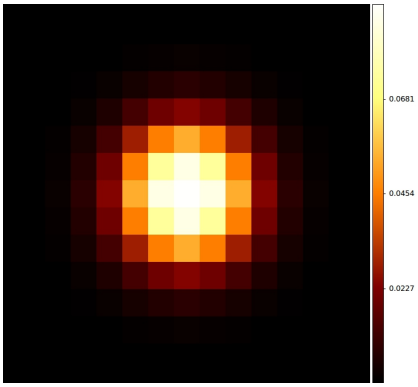
The PSFs created in section 7.4 are taken and all the missing data pixels are taken care by by replacing them by a value of 0 in the original image. This step ensures the PSF has properly distributed gaussian with no missing data at the boundaries.



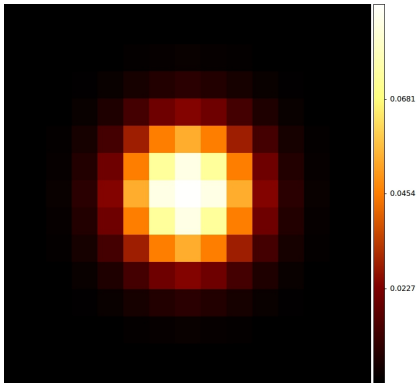
(a) PSF - 3000 GHz



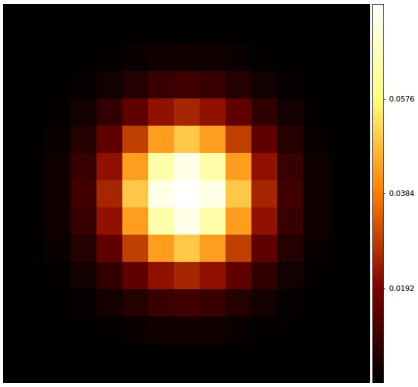
(b) PSF - 857 GHz



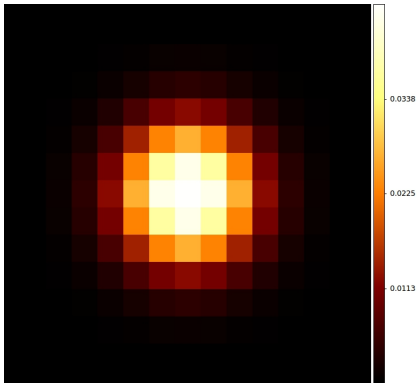
(c) PSF - 545 GHz



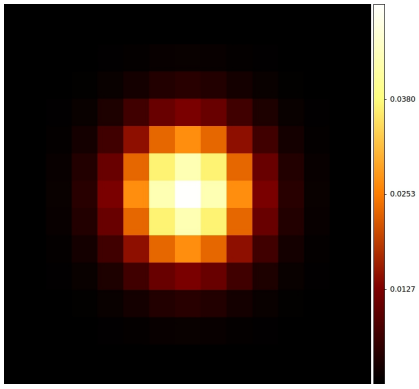
(d) PSF - 353 GHz



(e) PSF - 217 GHz



(f) PSF - 143 GHz



(g) PSF target

Fig. 7.4 Gaussian PSFs generated for detectors of different frequencies

Rotate PSF

There are different types of PSF and they may not be aligned. In that case, we need to rotate the PSF so as to get it aligned with image. The PSF considered here is a gaussian. So, there's no need to rotate the PSF.

Normalize the PSF

At this point, PSF is normalized by dividing the PSF gaussian matrix by its sum. This ensures that by convolving the further generated kernels with the images, images are actually blurred.

Resample high resolution images

Each of the PSF, after generation, have different dimensions. Since all the PSFs are to be converted to a reference PSF, it's required to resample them to the dimensions of reference PSF. It's also necessary to keep the dimensions of the kernel odd so that when we can convolve images with kernels at each pixel.

Trim or Zero-pad

After resampling of the PSF, the source PSF has either larger or smaller size compared to reference PSF. If the source PSF has more number of pixels, it's trimmed to the shape of target PSF. In the other case, where source PSF shape is smaller compared to target PSF, then missing data pixels are replaced with 0s.

Optical transfer function

Compute the Fast Fourier Transform (FFT) of the point-spread function (PSF) array and creates the optical transfer function (OTF) array that is not influenced by the PSF off-centering. By default, the OTF array is the same size as the PSF array.

To ensure that the OTF is not altered due to PSF off-centering, It is required to post-pad the PSF array (down or to the right) with zeros to match dimensions, then circularly shifts the values of the PSF array up (or to the left) until the central pixel reaches (1,1) position.

Wiener filter

Create a Wiener filter using a PSF image. The signal is ℓ_2 penalized by a 2D Laplacian operator that serves as a high-pass filter for the regularization process. The key to the process

is to use optical transfer functions (OTF) instead of simple Fourier transform, since it ensures the phase of the PSF is adequately placed.

2D Laplacian Operator :

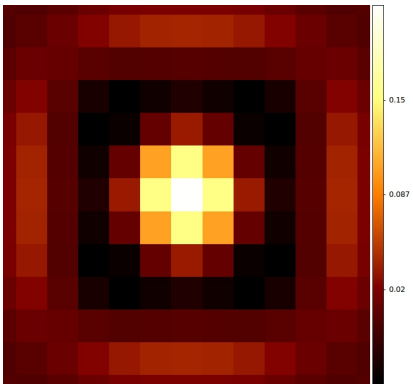
$$\begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

Compute Homogenization kernel

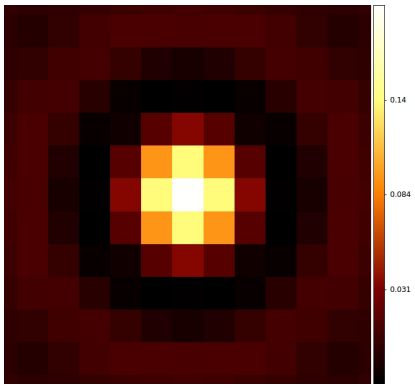
Compute the homogenization kernel to match two PSFs. The deconvolution step is done using a Wiener filter with ℓ_2 penalization. Source PSF is passed through Weiner filter first and then it's multiplied with Fast Fourier Transform (FFT) of target PSF. Next, the real values of inverse Fourier transform of the result are taken. This is the final kernel output. The output is given both in Fourier and in the image domain to serve different purposes.

Convolution of Images

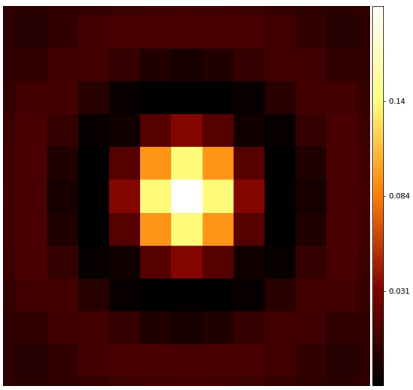
The data images after the unit conversion are now convolved with the kernels generated in section 7.5. The following convoluted images are obtained:



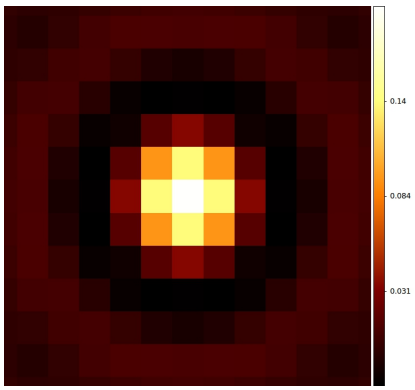
(a) Kernel - 3000 GHz



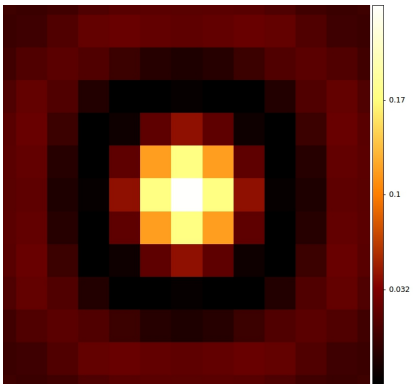
(b) Kernel - 857 GHz



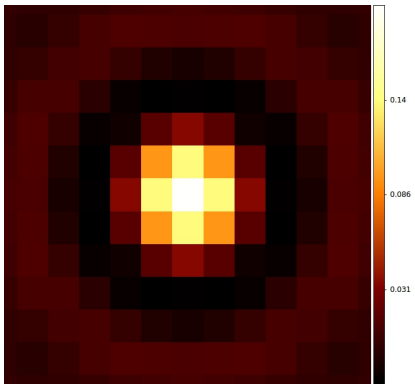
(c) Kernel - 545 GHz



(d) Kernel - 353 GHz

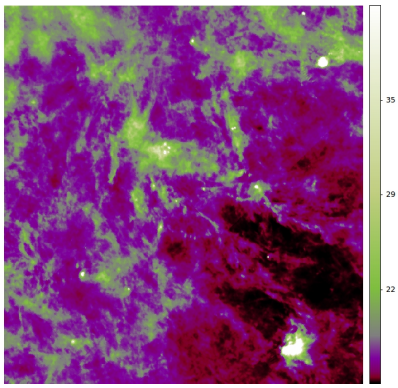


(e) Kernel - 217 GHz

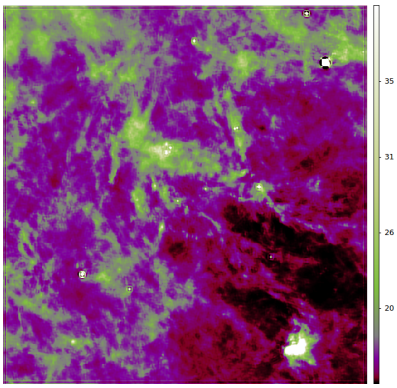


(f) Kernel - 143 GHz

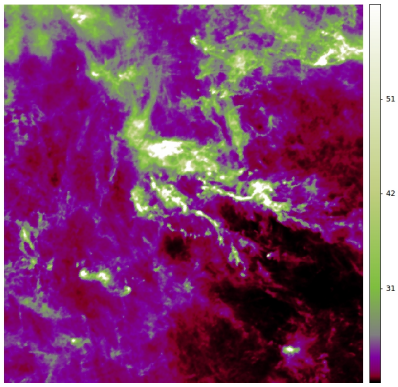
Fig. 7.5 Kernels generated for detectors of different frequecies



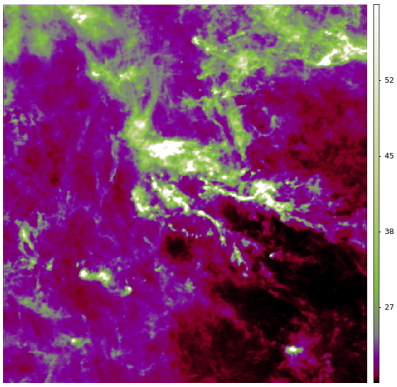
(a) Original - 3000 GHz



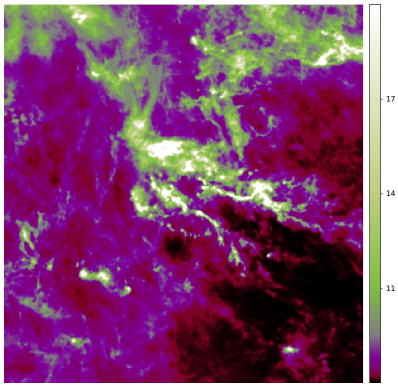
(b) Convoluted - 3000 GHz



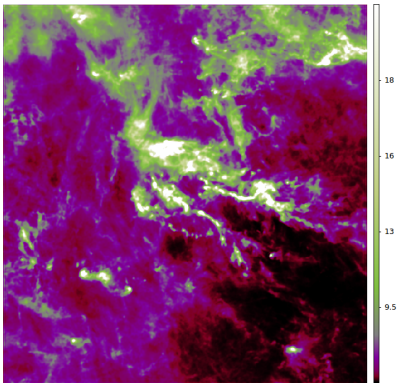
(c) Original - 857 GHz



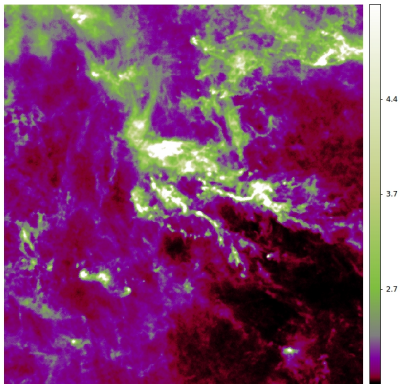
(d) Convoluted - 857 GHz



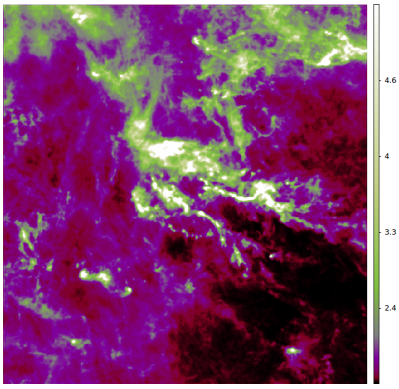
(e) Original - 545 GHz



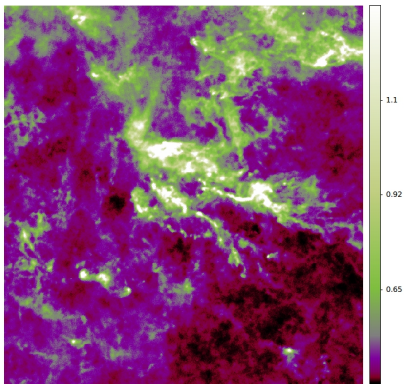
(f) Convoluted - 545 GHz



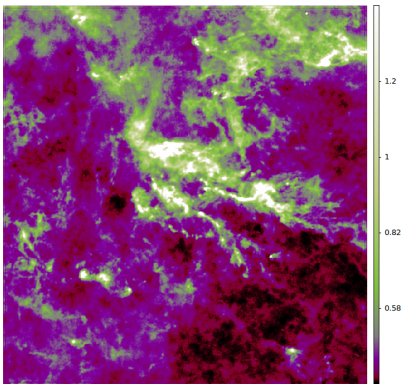
(g) Original - 353 GHz



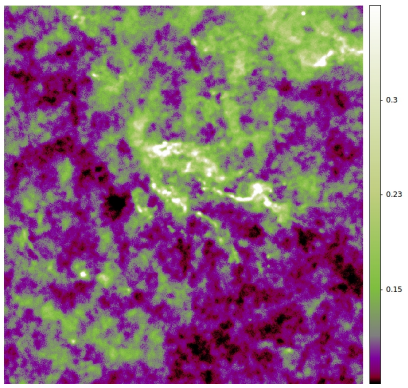
(h) Convoluted - 353 GHz



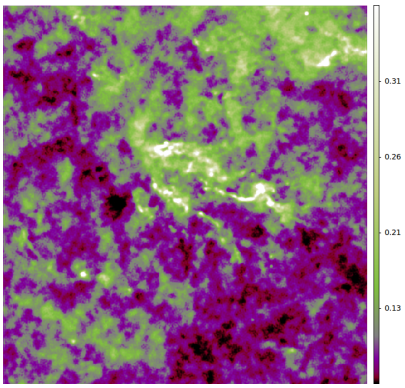
(i) Original - 217 GHz



(j) Convoluted - 217 GHz



(k) Original - 143 GHz



(l) Convoluted - 143 GHz

Fig. 7.6 Changes in images before and after convolution with the generated kernels

Chapter 8

Approach

8.1 Emission spectrum of thermal dust

8.1.1 Reference spectrum / Background correction

The combined data from IRAS and *Planck* gives the spectral energy distribution (SED) for each of the pixel in the images of bands from 3000 GHz to 100 GHz. The CMB has been removed from the images being used. But there might be some galactic and non-galactic emissions in the images that are not associated with the molecular cloud considered. Therefore, a reference window of 17'*17' is chosen from the faintest region from all the images and the average brightness of this window is subtracted.

8.1.2 Choice of spectral bands

In this project, our focus is to analyse the emission of dust particles between thermal emission and absorption of UV and visible photons from incident radiation. Therefore, we are not interested in the IRAS maps at 12, 25, 60 μm. Because everytime there's an absorption of UV/visible photon, small dust particlaes are heated and they contribute to the emission observed. Also when we use 100 GHz and 217 GHz bands, we see significant excess in the I_ν values caused by ^{12}CO and ^{13}CO emissions. Therefore, the frequency bands we use for generating the temperature plot are 3000(100 μm), 857, 545, 353, 217, 143 GHz.

8.2 Methods

In this process of finding the temperatur plot, our main aim is to generate equations in three parameters T_d, τ_ν and β and then fit the modified blackbody curve. There are two methods that we can follow to achieve this:

1. Pixel to Pixel data fitting
2. I_ν equation ratio method

In this project, we follow pixel to pixel data fitting method to obtain the temperature, optical depth and spectral emissivity index plot of the molecular clouds.

8.2.1 Principle of SED fitting

The data fitting technique used here is similar to the one used in Lombardi et al.2014[23]. The dust observed in the molecular clouds is optically thin at the observation frequencies we are considering. Therefore, its emission can be modelled as a modified black body[24]:

$$I_\nu = B_\nu(T)(1 - e^{\tau_\nu}) \simeq B_\nu(T)\tau_\nu \tag{8.1}$$

where, $B_\nu(T)$ is the black body function at temperature T and optical thickness τ_ν is the power law of frequency ν,

B_ν is given as,

$$B_\nu(T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{k_B T_d}} - 1} \tag{8.2}$$

and τ_ν as,

$$\tau_\nu = \tau_{\nu 0} \left(\frac{\nu}{\nu_0} \right)^\beta \tag{8.3}$$

The frequency ν_0 is an arbitrary frequency we set. Here we set ν_0 = 1200 GHz[25].

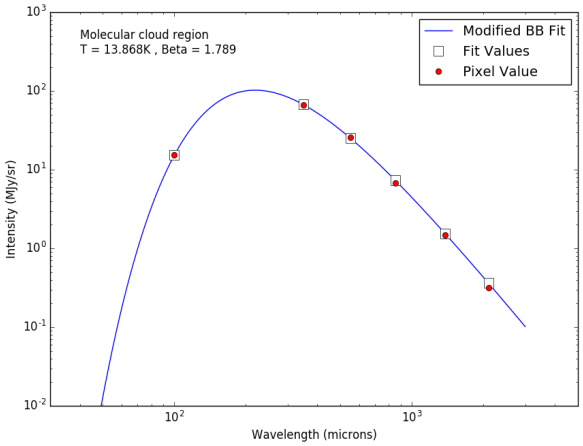
We need to convert the measured flux into intensity. Also the beam size at the specific wavelength must be considered. Here, we make an assumption that temperature gradients are negligible along the line of sight. This is an approximation we make as we know at low temperatures that characterize molecular clouds, even the small change in temperature causes large increase in the intensity. Therefore, if there's gradient present along line of sight, we receive photons from the warmer regions mostly because of which, the temperature obtained from the fit will be biased high. This will cause in false calculations of optical depth. This effect increases as the gradient increases. For these regions, we consider T in equation 8.1 as *effective dust temperature*. [23]

In the process of fitting the data from IRAS and *Planck*, we consider a pixel position and take the I_ν value of the same pixel for the images in all the bands. Now, we have 6 data points available to us for blackbody curve fitting. From the equation 8.1, 8.2, 8.3, we have three parameters T_d, τ_ν and β. And to find these parameters we generate 6 equations. Using these 6 equations, we fit a least square model to the parameters starting the iterations for fitting process with initial values for the parameters. We get values for temperature, optical depth and spectral emissivity index at the pixel considered in all the images. Similarly, we can continue this process spanning over entire image to get these values and generate the plot for each of the parameter. The detailed fitting process is explained in Lombardi et. al. 2014[23].

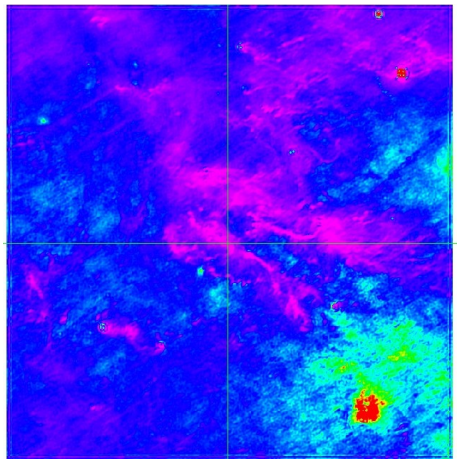
8.3 Example of spectra

After the data fitting of the images has been done, we plot for modified blackbody radiation to know the temperature at each pixel. Plots are shown for two pixels from Taurus molecular cloud (one each in molecular and non-molecular region) in figure 8.1 and 8.2.

The two spectra shown in these figures, shows that a single modified blackbody gives a representation of SED measured by IRAS and *Planck*. By increasing the number of free parameters, we can improve the fit significantly.

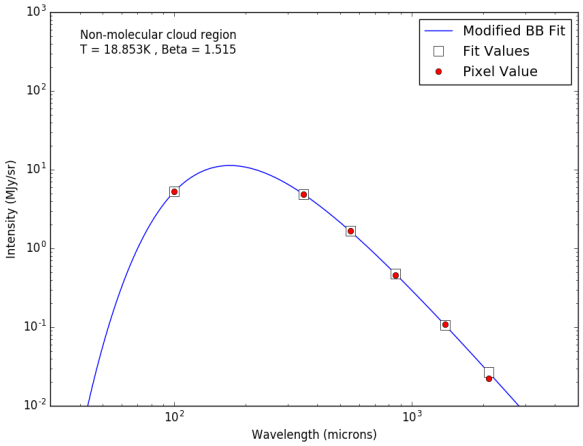


(a)

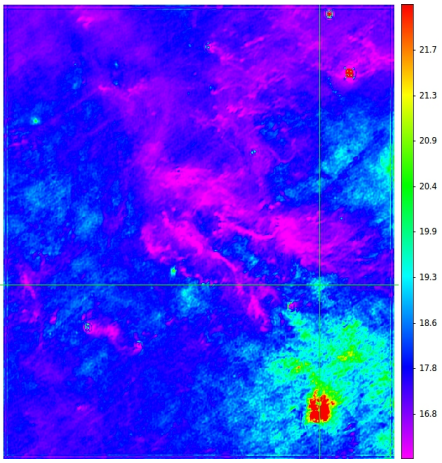


(b)

Fig. 8.1 In figure (a), The plot shows the modified blackbody curve fit for a location in molecular region of the Taurus molecular cloud. Red dots are the pixel values, boxes are fitted model integrated within the bands, and the solid line is the fitted model. In figure (b), The location of the pixel is shown in the molecular cloud.



(a)



(b)

Fig. 8.2 In figure (a), The plot shows the modified blackbody curve fit for a location in non-molecular region of the Taurus molecular cloud. Red dots are the pixel values, boxes are fitted model integrated within the bands, and the solid line is the fitted model. In figure (b), The location of the pixel is shown in the molecular cloud.

Chapter 9

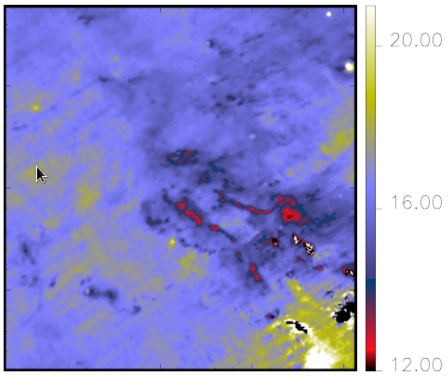
Results and Analysis

9.1 Validation

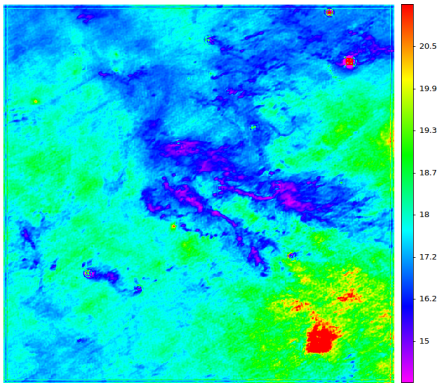
The convoluted images of the cloud in all bands were obtained as explained in chapter 7. Later, we fitted the data from each pixel in all images as followed in chapter 8. For our analysis, we used Taurus molecular cloud as our reference to validate the method to generate the temperature plots using our python code. For the validation, we used an existing study on the chosen molecular cloud[25]. The measured spectra are fitted to a single modified blackbody. It can be seen that the results fit reasonably. The plots for temperature, optical depth (τ), spectral emissivity index (β) are plotted. Also $T - \beta$ anti-correlation can be clearly seen from the plot. Histogram plots of temperature and beta were also generated.

9.1.1 Dust temperature map

The dust temperature map is plotted in the figure 9.1. Different structures of the Taurus molecular cloud can clearly be seen in the figure. Temperature range for both reference and obtained plots is also similar.



(a) Reference temperature plot[25]



(b) Obtained temperature plot

Fig. 9.1 Dust temperature plots (Kelvin)

9.1.2 Optical Depth map

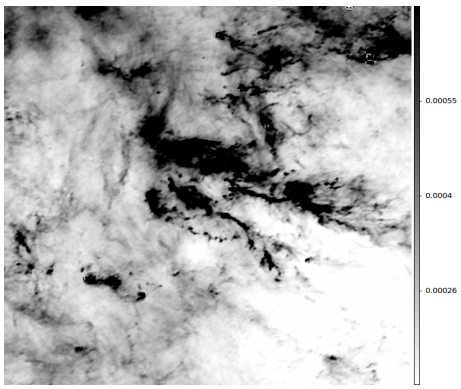
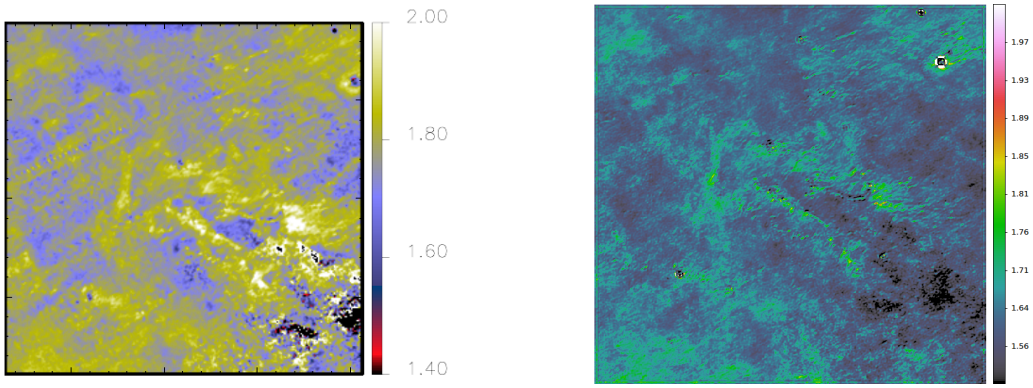


Fig. 9.2 Optical depth map of taurus molecular cloud

9.1.3 Spectral emissivity index map

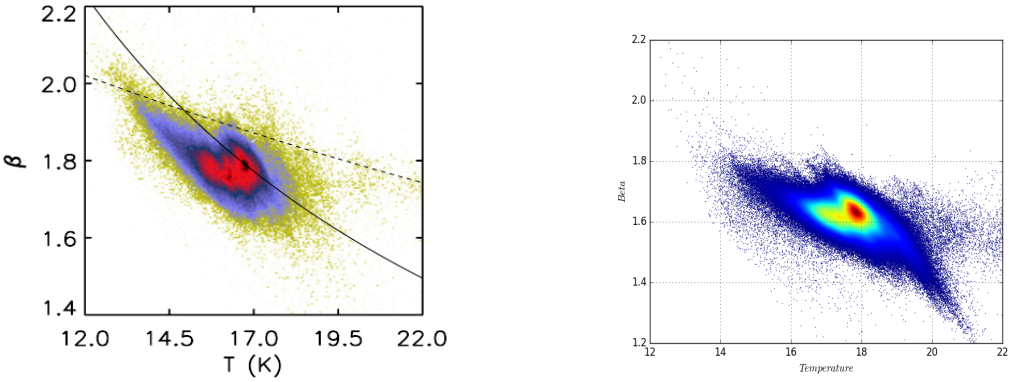


(a) Reference β plot[25] (b) Obtained β plot

Fig. 9.3 Spectral emissivity index (β) plot

9.1.4 Detailed analysis

Correlation plots

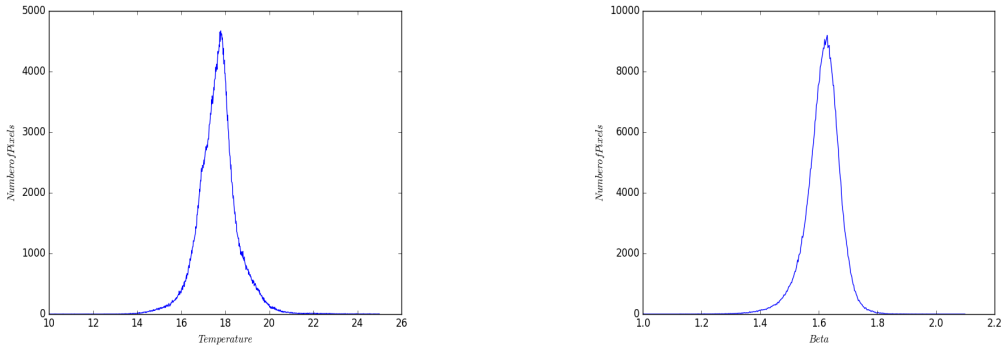


(a) Reference plot[25] (b) Obtained plot

Fig. 9.4 Correlation between T and β in the maps

Histograms

Figure 9.5 shows the relative distribution of temperature (T) and spectral emissivity index (β).



(a) T (b) β

Fig. 9.5 Pixel distribution plots of T and β

Applying the approach on test data

9.2 λ Orionis

Lambda Orionis[26] is a hot, massive star that is surrounded by several other hot, massive stars, all of which are creating radiation that excites a ring of dust, creating the "Lambda Orionis molecular ring, located at a distance of 1060 light years. " Also known as SH 2-264, the Lambda Orionis molecular ring is sometimes called the Meissa ring. In Arabic, the star Lambda Orionis is known as "Meissa" or "Al-Maisan," meaning "the shining one." The Meissa Ring is of interest to astronomers because it contains clusters of young stars and proto-stars, or forming stars, embedded within the clouds. With a diameter of approximately 130 light-years, the Lambda Orionis molecular ring is notable for being one of the largest star-forming regions WISE[27] has seen.

Data:

Name: λ Orionis

Reference longitude: 195.051°

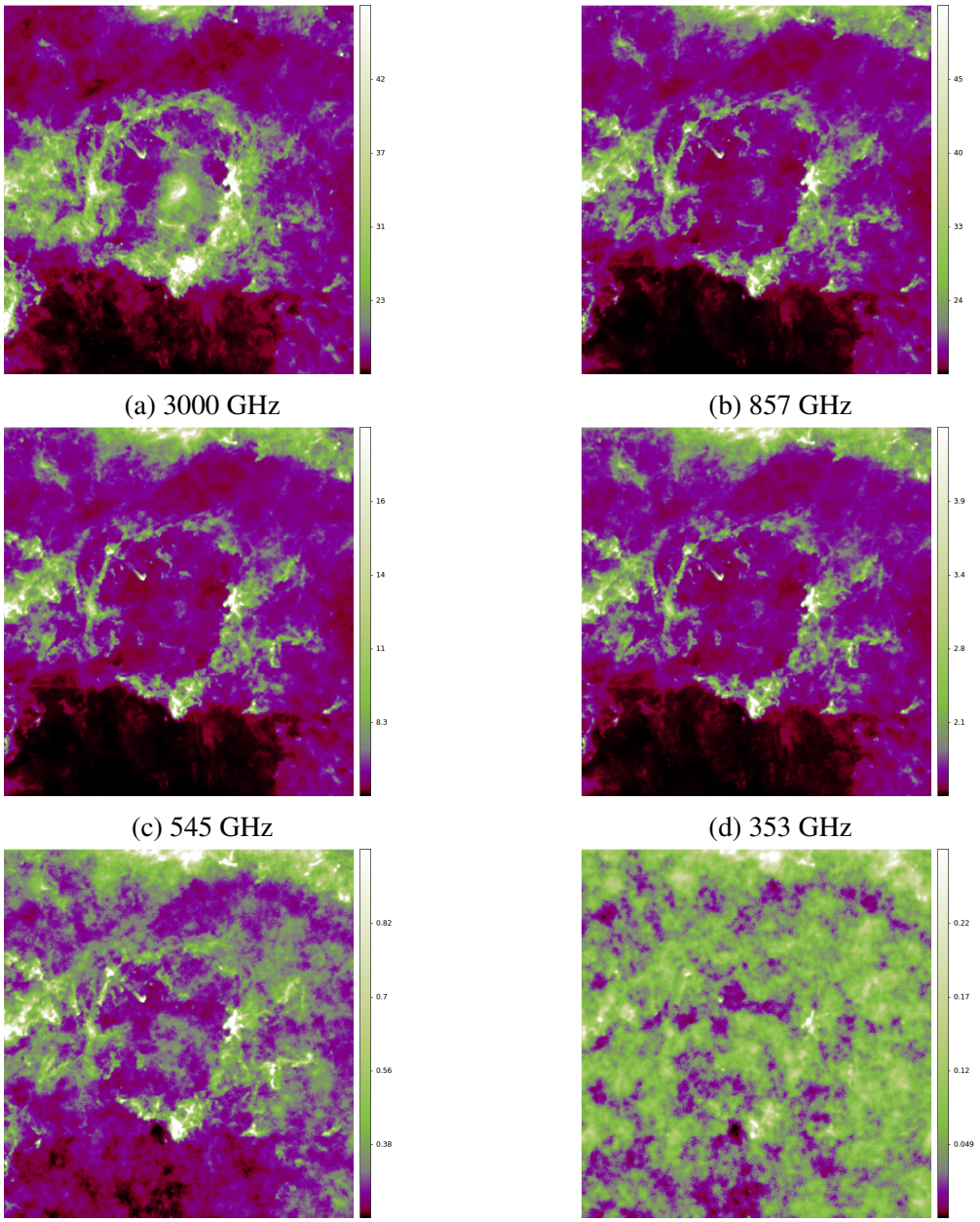
Reference latitude: -11.995°

Dimension of the image: 700*700

Image size (degrees): 19.833

Pixel scale: 1.7’ (including iris 100 image)

For all bands, we obtain images with size 19.833 °*19.833 ° window centered at l = 195.051 ° and b = -11.995 °



(a) 3000 GHz (b) 857 GHz

(c) 545 GHz (d) 353 GHz

(e) 217 GHz (f) 143 GHz

Fig. 9.6 IRAS and HFI maps of λ Orionis, 143, 217, 353, 545, 857, 3000 GHz in MJy/Sr (with conversion)

The above images are the data maps from IRAS and *Planck*.

9.2.1 Dust Temperature map

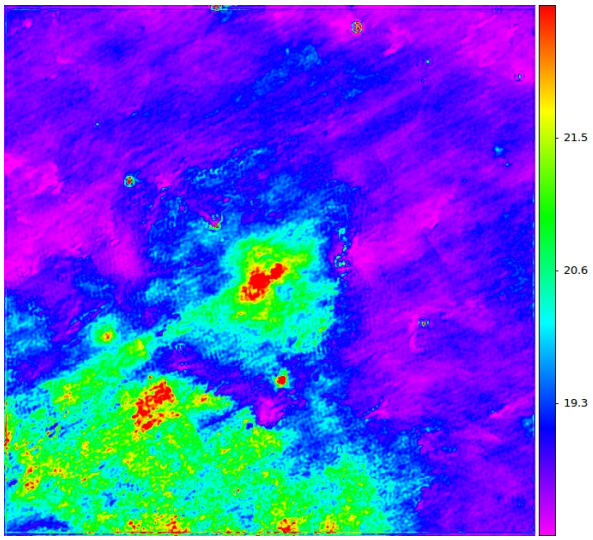


Fig. 9.7 Dust temperature map of λ Orionis

9.2.2 Spectral emissivity index map

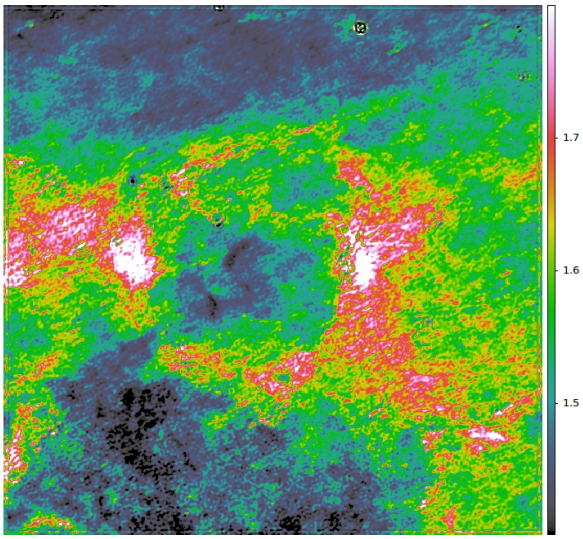


Fig. 9.8 Spectral emissivity index map of λ Orionis

9.2.3 Optical depth map

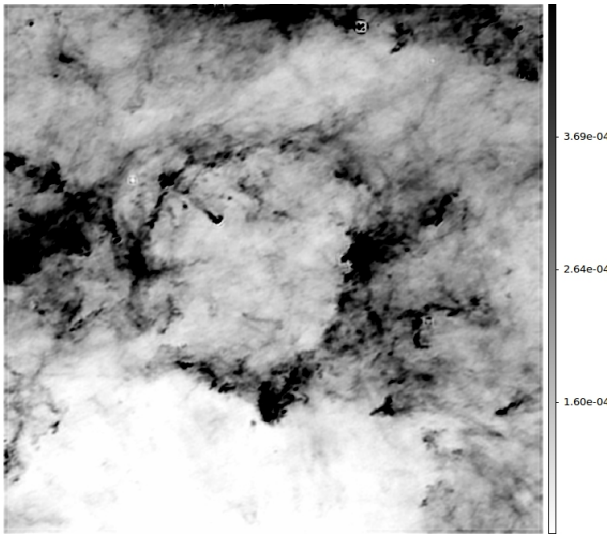
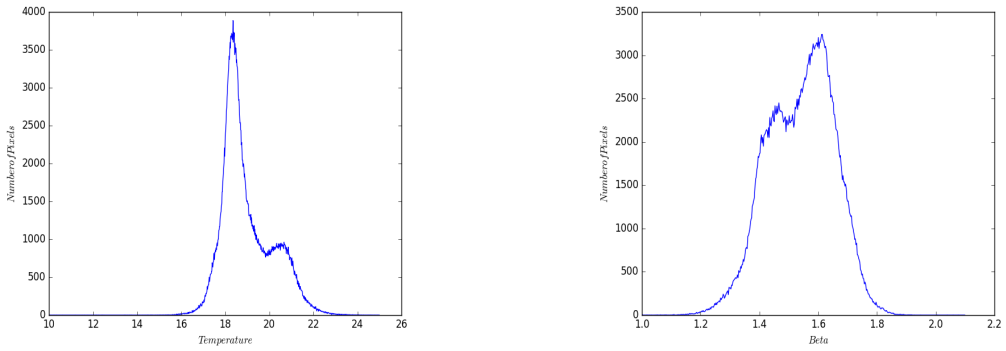


Fig. 9.9 Optical depth map of λ Orionis

9.2.4 Detailed Analysis

Histograms



(a) T (b) β

Fig. 9.10 Pixel distribution plots of T and β

Correlation

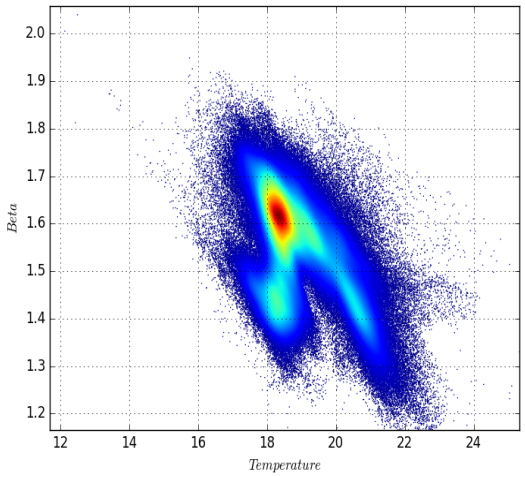


Fig. 9.11 Correlation between T and β

9.3 Perseus molecular cloud

The Perseus molecular cloud (Per MCld)[24] is a nearby (600 ly) giant molecular cloud in the constellation of Perseus and contains over 10,000 solar masses of gas and dust covering an area of 6 by 2 degrees in the sky. Unlike the Orion molecular cloud it is almost invisible apart from two clusters, IC 348 and NGC 1333, where low-mass stars are formed. It is very bright at mid and far-infrared wavelengths and in the submillimeter originating in dust heated by the newly formed low-mass stars.

Data:

Name: Perseus Molecular cloud

Reference longitude: 159.450°

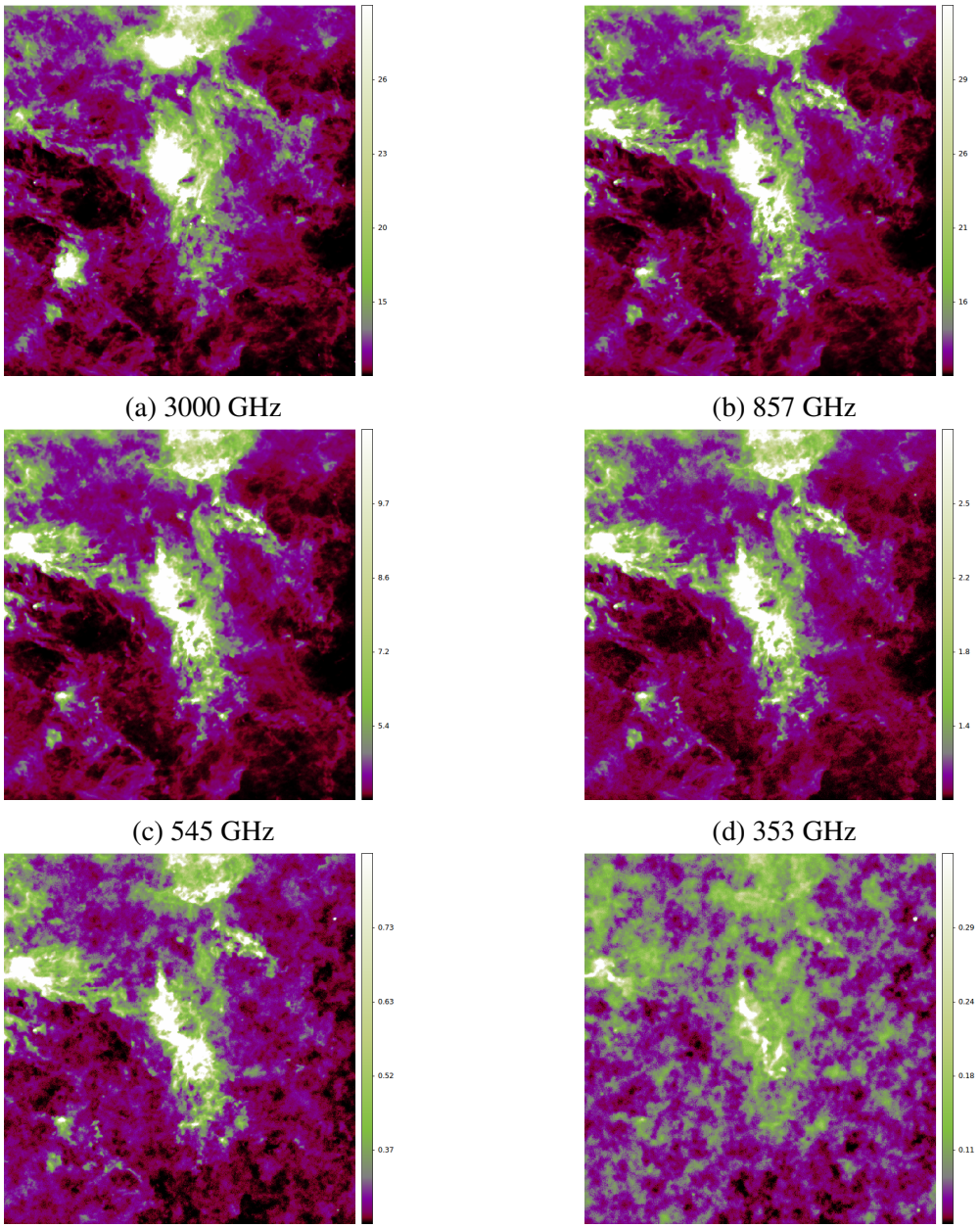
Reference latitude: -19.819°

Dimension of the image: 700*700

Image size (degrees): 19.833

Pixel scale: 1.7’ (including iris 100 image)

For all bands, we obtain images with size 19.833 °*19.833 ° window centered at l = 159.450 ° and b = -19.819 °



(a) 3000 GHz (b) 857 GHz

(c) 545 GHz (d) 353 GHz

(e) 217 GHz (f) 143 GHz

Fig. 9.12 IRAS and HFI maps of Perseus molecular cloud, 143, 217, 353, 545, 857, 3000 GHz in MJy/Sr (with conversion)

9.3.1 Dust Temperature map

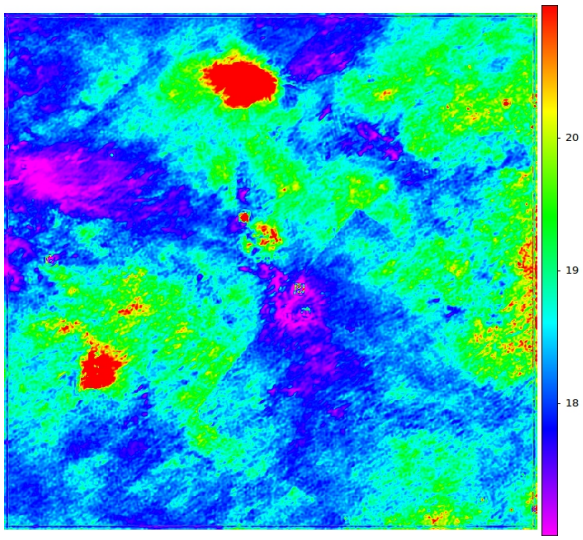


Fig. 9.13 Dust temperature map of Perseus molecular cloud

9.3.2 Spectral emissivity index map

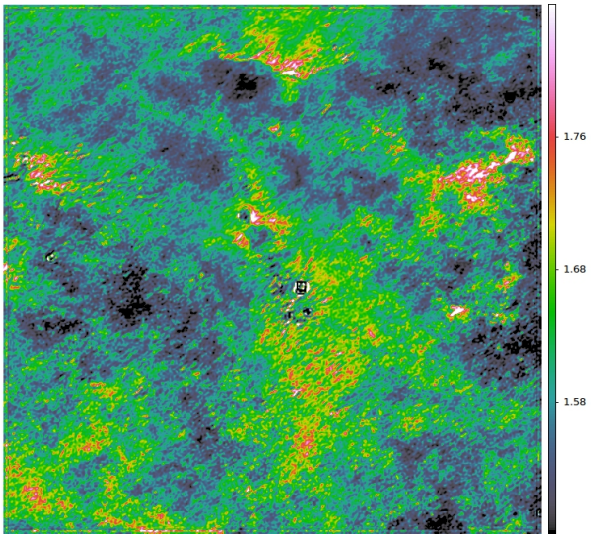


Fig. 9.14 Spectral emissivity index map of Perseus molecular cloud

9.3.3 Optical depth map

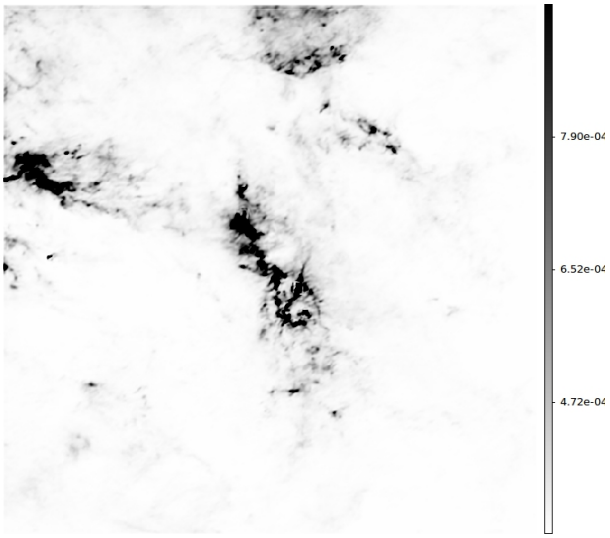
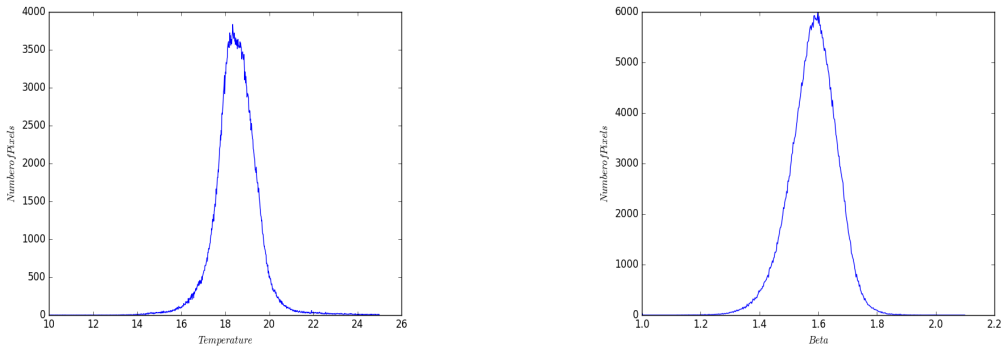


Fig. 9.15 Optical depth map of Perseus molecular cloud

9.3.4 Detailed Analysis

Histograms



(a) T (b) β

Fig. 9.16 Pixel distribution plots of T and β

Correlation

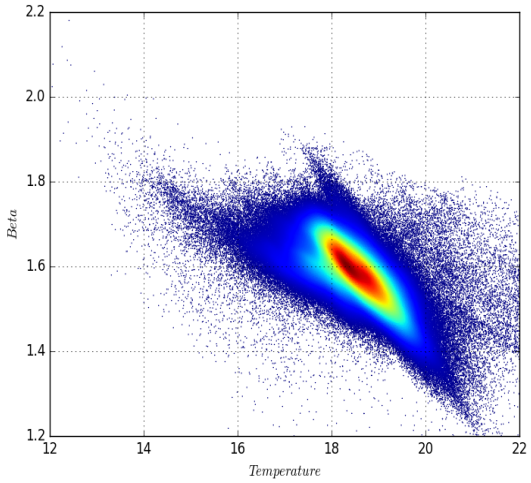


Fig. 9.17 Correlation between T and β

Chapter 10

Conclusions

1. It's observed from the temperature and spectral emissivity index plots that higher the temperature, lower the region density. Hence we can say, denser regions are colder compared to low density regions. We also see abrupt changes in temperature at few locations. This is due to young steller object (YSO) sources present in the molecular cloud.

2. In the dust temperature maps, it can be seen that the cooling patterns in the molecular cloud varies from 14 K to 21 K in case of Taurus molecular cloud, 16 K to 27 K in case of λ Ori and 14 K to 26 K in case of Perseus molecular cloud.

3. There's variation in spectral emissivity index also. For Taurus molecular cloud, β ranges from 1.6 to 1.9, where as for λ Orionis, it varies from 1.4 to 1.8. And in case of Perseus molecular cloud, the variation is from 1.5 to 1.85.

4. From the anti-correlation between T and β, it can be seen that colder the region in the molecular cloud, higher the values of β.

5. Another important observation from the analysis - spatial distribution of column density of molecular cloud from dense to faint regions is clearly seen in the optical depth map.

REFERENCES

[1] "Iras image."

[2] "Point spread function."

[3] P. Collaboration, P. Ade, N. Aghanim, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. Banday, R. Barreiro, N. Bartolo, S. Basak, P. Battaglia, E. Battaner, K. Benabed, A. Benoît, A. Benoit-Lévy, J. P. Bernard, M. Bersanelli, P. Bielewicz, J. Bock, A. Bonaldi, L. Bonavera, J. Bond, J. Borrill, F. Bouchet, M. Bucher, C. Burigana, R. Butler, E. Calabrese, J. F. Cardoso, G. Castex, A. Catalano, A. Chamballu, P. Christensen, S. Colombi, L. Colombo, B. Crill, A. Curto, F. Cuttaia, L. Danese, R. Davies, R. Davis, d. P, Bernardis, d. A, Rosa, d. G, Zotti, J. Delabrouille, C. Dickinson, J. Diego, H. Dole, S. Donzelli, O. Doré, M. Douspis, A. Ducout, X. Dupac, G. Efstathiou, F. Elsner, T. A. Ensslin, H. Eriksen, J. Fergusson, F. Finelli, O. Forni, M. Frailis, C. Franceschet, E. Franceschi, A. Frejsel, S. Galeotta, S. Galli, K. Ganga, M. Giard, Y. Giraud-Héraud, E. Gjerlow, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gregorio, A. Gruppuso, F. Hansen, D. Hanson, D. Harrison, S. Henrot-Versillé, D. Herranz, S. Hildebrandt, E. Hivon, M. Hobson, W. Holmes, A. Hornstrup, W. Hovest, K. Huffenberger, G. Hurier, A. Jaffe, T. Jaffe, M. Juvela, E. Keihänen, R. Keskitalo, K. Kiiveri, T. Kisner, J. Knoche, N. Krachmalnicoff, M. Kunz, and H. Kurki-Suonio, "Planck 2015 results. ii. low frequency instrument data processing," *A&A*, feb 2015.

[4] P. Collaboration, "Planck 2015 results. viii. high frequency instrument data processing: Calibration and maps," *A&A*, sep 2015.

[5] "3.6 meter optical telescope project at devasthal."

[6] R. Sagar, B. Kumar, and A. Omar, "Optical astronomical facilities at nainital, india." 2013.

[7] S. B. Howell, "Photometry and astrometry," in *Handbook of CCD Astronomy*, p. 105, Cambridge University Press, 2nd ed., 2006.

[8] R. B. Larson, "The evolution of molecular clouds," in *The Structure and Content of Molecular Clouds 25 Years of Molecular Radioastronomy* (T. L. Wilson and K. J. Johnston, eds.), pp. 13–28, Berlin, Heidelberg: Springer Berlin Heidelberg, 1994.

[9] S. L. Schnee, "Gas and dust in molecular clouds : Density, temperature and velocity structure." 2006.

[10] P. Collaboration, P. A. R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi, M. Baker, A. Balbi, A. J. Banday, R. B. Barreiro, J. G. Bartlett, E. Battaner, K. Benabed, K. Bennett, A. Benoît, J.-P. Bernard, M. Bersanelli, R. Bhatia, J. J. Bock, A. Bonaldi, J. R. Bond, J. Borrill, F. R. Bouchet, T. Bradshaw, M. Bremer, M. Bucher, C. Burigana, R. C. Butler, P. Cabella, C. M. Cantalupo, B. Cappellini, J.-F. Cardoso, R. Carr, M. Casale, A. Catalano, L. Cayón, A. Challinor, A. Chamballu, J. Charra, R.-R. Chary, L.-Y. Chiang, C. Chiang, P. R. Christensen, D. L. Clements, S. Colombi, F. Couchot, A. Coulais, B. P. Crill, G. Crone, M. Crook, F. Cuttaia, L. Danese, O. D'Arcangelo, R. D. Davies, R. J. Davis, P. de Bernardis, J. de Bruin, G. de Gasperis, A. de Rosa, G. de Zotti, J. Delabrouille, J.-M. Delouis, F.-X. Désert, J. Dick, C. Dickinson, K. Dolag, H. Dole, S. Donzelli, O. Doré, U. Dörl, M. Douspis, X. Dupac, G. Efstathiou, T. A. Ensslin, H. K. Eriksen, F. Finelli, S. Foley, O. Forni, P. Fosalba, M. Frailis, E. Franceschi, M. Freschi, T. C. Gaier, S. Galeotta, J. Gallegos, B. Gandolfo, K. Ganga, M. Giard, G. Giardino, G. Gienger, Y. Giraud-Héraud, J. González, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gregorio, A. Gruppuso, G. Guyot, J. Haissinski, F. K. Hansen, D. Harrison, G. Helou, S. Henrot-Versillé, C. Hernández-Monteagudo, D. Herranz, S. R. Hildebrandt, E. Hivon, M. Hobson, W. A. Holmes, A. Hornstrup, W. Hovest, R. J. Hoyland, K. M. Huffenberger, A. H. Jaffe, T. Jagemann, W. C. Jones, J. J. Juillet, M. Juvela, P. Kangaslahti, E. Keihänen, R. Keskitalo, T. S. Kisner, R. Kneissl, L. Knox, M. Krassenburg, H. Kurki-Suonio, G. Lagache, A. Lähteenmäki, J.-M. Lamarre, A. E. Lange, A. Lasenby, R. J. Laureijs, C. R. Lawrence, S. Leach, J. P. Leahy, R. Leonardi, C. Leroy, P. B. Lilje, M. Linden-Vornle, M. López-Caniego, S. Lowe, P. M. Lubin, J. F. Macías-Pérez, T. Maciaszek, C. J. MacTavish, B. Maffei, D. Maino, N. Mandolesi, R. Mann, M. Maris, E. Martínez-González, S. Masi, M. Massardi, S. Matarrese, F. Matthai, P. Mazzotta, A. McDonald, P. McGehee, P. R. Meinhold, A. Melchiorri, J.-B. Melin, L. Mendes, A. Mennella, C. Mevi, R. Miniscalco, S. Mitra, M.-A. Miville-Deschênes, A. Moneti, L. Montier, G. Morgante, N. Morisset, D. Mortlock, D. Munshi, A. Murphy, P. Naselsky, P. Natoli, C. B. Netterfield, H. U. Norgaard-Nielsen, F. Noviello, D. Novikov, I. Novikov, I. J. O'Dwyer, I. Ortiz, S. Osborne, P. Osuna, C. A. Oxborrow, F. Pajot, R. Paladini, B. Partridge, F. Pasian, T. Passvogel, G. Patanchon, D. Pearson, T. J. Pearson, O. Perdereau, L. Perotto, F. Perrotta, F. Piacentini, M. Piat, E. Pierpaoli, S. Plaszczynski, P. Platania, E. Pointecouteau, G. Polenta, N. Ponthieu, L. Popa, T. Poutanen, G. Prézeau, S. Prunet, J.-L. Puget, J. P. Rachen, W. T. Reach, R. Rebolo, M. Reinecke, J.-M. Reix, C. Renault, S. Ricciardi, T. Riller, I. Ristorcelli, G. Rocha, C. Rosset, M. Rowan-Robinson, J. A. Rubiño-Martín, B. Rusholme, E. Salerno, M. Sandri, D. Santos, G. Savini, B. M. Schaefer, D. Scott, M. D. Seiffert, P. Shellard, A. Simonetto, G. F. Smoot, C. Sozzi, J.-L. Starck, J. Sternberg, F. Stivoli, V. Stolyarov, R. Stompor, L. Stringhetti, R. Sudiwala, R. Sunyaev, J.-F. Sygnet, D. Tapiador, J. A. Tauber, D. Tavagnacco, D. Taylor, L. Terenzi, D. Texier, L. Toffolatti, M. Tomasi, J.-P. Torre, M. Tristram, J. Tuovinen, M. Türler, M. Tuttlebee, G. Umana, L. Valenziano, J. Valiviita, J. Varis, L. Vibert, P. Vielva, F. Villa, N. Vittorio, L. A. Wade, B. D. Wandelt, C. Watson, S. D. M. White, M. White, A. Wilkinson, D. Yvon, A. Zacchei, and A. Zonca, "Planck early results. i. theplanck mission," *A&A*, vol. 536, p. A1, dec 2011.

[11] "Planck satellite."

[12] "Planck satellite image."

[13] "Model of the planck satellite."

[14] M. MivilleDeschenes and G. Lagache, "Iris: A new generation of iras maps," *ASTROPHYS J SUPPL S*, vol. 157, pp. 302–323, apr 2005.

[15] "Nssdca/cospar id: 1983-004a."

[16] "Skyview - virtual observatory (query form)."

[17] P. Collaboration, P. A. R. Ade, N. Aghanim, C. Armitage-Caplan, M. Arnaud, M. Ashdown, F. Atrio-Barandela, J. Aumont, C. Baccigalupi, A. J. Banday, R. B. Barreiro, E. Battaner, K. Benabed, A. Benoît, A. Benoit-Lévy, J.-P. Bernard, M. Bersanelli, P. Bielewicz, J. Bobin, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bridges, M. Bucher, C. Burigana, J.-F. Cardoso, A. Catalano, A. Challinor, A. Chamballu, R.-R. Chary, X. Chen, H. C. Chiang, L.-Y. Chiang, P. R. Christensen, S. Church, D. L. Clements, S. Colombi, L. P. L. Colombo, C. Combet, B. Comis, F. Couchot, A. Coulais, B. P. Crill, A. Curto, F. Cuttaia, L. Danese, R. D. Davies, P. de Bernardis, A. de Rosa, G. de Zotti, J. Delabrouille, J.-M. Delouis, F.-X. Désert, C. Dickinson, J. M. Diego, H. Dole, S. Donzelli, O. Doré, M. Douspis, X. Dupac, G. Efstathiou, T. A. Ensslin, H. K. Eriksen, E. Falgarone, F. Finelli, O. Forni, M. Frailis, E. Franceschi, S. Galeotta, K. Ganga, M. Giard, Y. Giraud-Héraud, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gregorio, A. Gruppuso, F. K. Hansen, D. Hanson, D. Harrison, S. Henrot-Versillé, C. Hernández-Monteagudo, D. Herranz, S. R. Hildebrandt, E. Hivon, M. Hobson, W. A. Holmes, A. Hornstrup, W. Hovest, K. M. Huffenberger, G. Hurier, A. H. Jaffe, T. R. Jaffe, W. C. Jones, M. Juvela, E. Keihänen, R. Keskitalo, T. S. Kisner, R. Kneissl, J. Knoche, L. Knox, M. Kunz, H. Kurki-Suonio, G. Lagache, J.-M. Lamarre, A. Lasenby, R. J. Laureijs, C. R. Lawrence, J. P. Leahy, R. Leonardi, C. Leroy, J. Lesgourgues, M. Liguori, P. B. Lilje, M. Linden-Vornle, M. López-Caniego, P. M. Lubin, J. F. Macías-Pérez, B. Maffei, N. Mandolesi, M. Maris, D. J. Marshall, P. G. Martin, E. Martínez-González, S. Masi, M. Massardi, S. Matarrese, F. Matthai, P. Mazzotta, P. McGehee, A. Melchiorri, L. Mendes, A. Mennella, M. Migliaccio, S. Mitra, M.-A. Miville-Deschênes, A. Moneti, L. Montier, G. Morgante, D. Mortlock, D. Munshi, J. A. Murphy, P. Naselsky, F. Nati, P. Natoli, C. B. Netterfield, H. U. Norgaard-Nielsen, C. North, F. Noviello, D. Novikov, I. Novikov, S. Osborne, C. A. Oxborrow, F. Paci, L. Pagano, F. Pajot, D. Paoletti, F. Pasian, G. Patanchon, O. Perdereau, L. Perotto, F. Perrotta, F. Piacentini, M. Piat, E. Pierpaoli, D. Pietrobon, S. Plaszczynski, E. Pointecouteau, G. Polenta, N. Ponthieu, L. Popa, T. Poutanen, G. W. Pratt, G. Prézeau, S. Prunet, J.-L. Puget, J. P. Rachen, M. Reinecke, M. Remazeilles, C. Renault, S. Ricciardi, T. Riller, I. Ristorcelli, G. Rocha, C. Rosset, G. Roudier, B. Rusholme, D. Santos, G. Savini, D. Scott, E. P. S. Shellard, L. D. Spencer, J.-L. Starck, V. Stolyarov, R. Stompor, R. Sudiwala, F. Sureau, D. Sutton, A.-S. Suur-Uski, J.-F. Sygnet, J. A. Tauber, D. Tavagnacco, L. Terenzi, M. Tomasi, M. Tristram, M. Tucci, G. Umana, L. Valenziano, J. Valiviita, B. Van Tent, P. Vielva, F. Villa, N. Vittorio, L. A. Wade, B. D. Wandelt, D. Yvon, A. Zacchei, and A. Zonca, "Planck 2013 results. ix. hfi spectral response," *A&A*, vol. 571, p. A9, nov 2014.

[18] Rybicki, G. B. Lightman, and A. P, "Radiative processes in astrophysics," *Radiative Processes in Astrophysics, by George B. Rybicki, Alan P. Lightman, pp. 400. ISBN 0-471-82759-2. Wiley-VCH , June 1986.*, jun 1986.

[19] P. Collaboration, R. Adam, P. Ade, N. Aghanim, Y. Akrami, M. Alves, M. Arnaud, F. Arroja, J. Aumont, C. Baccigalupi, M. Ballardini, A. Banday, R. Barreiro, J. Bartlett, N. Bartolo, S. Basak, P. Battaglia, E. Battaner, R. Battye, K. Benabed, A. Benoît, A. Benoit-Lévy, J. P. Bernard, M. Bersanelli, B. Bertincourt, P. Bielewicz, A. Bonaldi, L. Bonavera, J. Bond, J. Borrill, F. Bouchet, F. Boulanger, M. Bucher, C. Burigana, R. Butler, E. Calabrese, J. F. Cardoso, P. Carvalho, B. Casaponsa, G. Castex, A. Catalano, A. Challinor, A. Chamballu, R. R. Chary, H. Chiang, J. Chluba, P. Christensen, S. Church, M. Clemens, D. Clements, S. Colombi, L. Colombo, C. Combet, B. Comis, D. Contreras, F. Couchot, A. Coulais, B. Crill, M. Cruz, A. Curto, F. Cuttaia, L. Danese, R. Davies, R. Davis, d. P, Bernardis, d. A, Rosa, d. G, Zotti, J. Delabrouille, J. M. Delouis, F. X. Désert, D. E, Valentino, C. Dickinson, J. Diego, K. Dolag, H. Dole, S. Donzelli, O. Doré, M. Douspis, A. Ducout, J. Dunkley, X. Dupac, G. Efstathiou, P. Eisenhardt, F. Elsner, T. A. Ensslin, H. Eriksen, E. Falgarone, Y. Fantaye, M. Farhang, S. Feeney, J. Fergusson, R. Fernandez-Cobos, F. Feroz, F. Finelli, E. Florido, O. Forni, M. Frailis, A. Fraisse, C. Franceschet, and E. Franceschi, "Planck 2015 results. i. overview of products and scientific results," *A&A*, feb 2015.

[20] R. Vio, J. Nagy, and W. Wamsteker, "Multiple-image deblurring with spatially-variant point spread functions," *A&A*, vol. 434, pp. 795–800, may 2005.

[21] G. Aniano, B. T. Draine, K. D. Gordon, and K. Sandstrom, "Common-resolution convolution kernels for space- and ground-based telescopes," *PUBL ASTRON SOC PAC*, vol. 123, pp. 1218–1236, oct 2011.

[22] B. Boucaud, "Psf homogenization kernels - pypher 0.6.1." 2016.

[23] M. Lombardi, H. Bouy, J. Alves, and C. J. Lada, "Herschel-planck dust optical-depth and column-density maps," *A&A*, vol. 566, p. A45, jun 2014.

[24] E. Zari, M. Lombardi, J. Alves, C. J. Lada, and H. Bouy, "Herschel-planck dust optical depth and column density maps," *A&A*, vol. 587, p. A106, mar 2016.

[25] P. Collaboration, A. Abergel, P. A. R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi, A. Balbi, A. J. Banday, R. B. Barreiro, J. G. Bartlett, E. Battaner, K. Benabed, A. Benoît, J.-P. Bernard, M. Bersanelli, R. Bhatia, J. J. Bock, A. Bonaldi, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bucher, C. Burigana, P. Cabella, J.-F. Cardoso, A. Catalano, L. Cayón, A. Challinor, A. Chamballu, L.-Y. Chiang, C. Chiang, P. R. Christensen, D. L. Clements, S. Colombi, F. Couchot, A. Coulais, B. P. Crill, F. Cuttaia, L. Danese, R. D. Davies, R. J. Davis, P. de Bernardis, G. de Gasperis, A. de Rosa, G. de Zotti, J. Delabrouille, J.-M. Delouis, F.-X. Désert, C. Dickinson, K. Dobashi, S. Donzelli, O. Doré, U. Dörl, M. Douspis, X. Dupac, G. Efstathiou, T. A. Ensslin, H. K. Eriksen, F. Finelli, O. Forni, M. Frailis, E. Franceschi, S. Galeotta, K. Ganga, M. Giard, G. Giardino, Y. Giraud-Héraud, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gregorio, A. Gruppuso, V. Guillet, F. K. Hansen, D. Harrison, S. Henrot-Versillé, D. Herranz, S. R. Hildebrandt, E. Hivon, M. Hobson, W. A. Holmes, W. Hovest, R. J. Hoyland, K. M. Huffenberger, A. H. Jaffe, A. Jones, W. C. Jones, M. Juvela, E. Keihänen, R. Keskitalo, T. S. Kisner, R. Kneissl, L. Knox, H. Kurki-Suonio, G. Lagache, J.-M. Lamarre, A. Lasenby, R. J. Laureijs, C. R. Lawrence, S. Leach, R. Leonardi, C. Leroy, M. Linden-Vornle, M. López-Caniego, P. M. Lubin, J. F. Macías-Pérez, C. J. MacTavish, B. Maffei, N. Mandolesi, R. Mann, M. Maris, D. J. Marshall, P. Martin,

E. Martínez-González, S. Masi, S. Matarrese, F. Matthai, P. Mazzotta, P. McGehee, P. R. Meinhold, A. Melchiorri, L. Mendes, A. Mennella, S. Mitra, M.-A. Miville-Deschênes, A. Moneti, L. Montier, G. Morgante, D. Mortlock, D. Munshi, A. Murphy, P. Naselsky, P. Natoli, C. B. Netterfield, H. U. Norgaard-Nielsen, F. Noviello, D. Novikov, I. Novikov, S. Osborne, F. Pajot, R. Paladini, F. Pasian, G. Patanchon, O. Perdereau, L. Perotto, F. Perrotta, F. Piacentini, M. Piat, S. Plaszczynski, E. Pointecouteau, G. Polenta, N. Ponthieu, T. Poutanen, G. Prézeau, S. Prunet, J.-L. Puget, W. T. Reach, R. Rebolo, M. Reinecke, C. Renault, S. Ricciardi, T. Riller, I. Ristorcelli, G. Rocha, C. Rosset, J. A. Rubiño-Martín, B. Rusholme, M. Sandri, D. Santos, G. Savini, D. Scott, M. D. Seiffert, P. Shellard, G. F. Smoot, J.-L. Starck, F. Stivoli, V. Stolyarov, R. Sudiwala, J.-F. Sygnet, J. A. Tauber, L. Terenzi, L. Toffolatti, M. Tomasi, J.-P. Torre, M. Tristram, J. Tuovinen, G. Umana, L. Valenziano, L. Verstraete, P. Vielva, F. Villa, N. Vittorio, L. A. Wade, B. D. Wandelt, D. Yvon, A. Zacchei, and A. Zonca, "Planck early results. xxv. thermal dust in nearby molecular clouds," *A&A*, vol. 536, p. A25, dec 2011.

[26] H. Bouy, N. Huélamo, D. Barrado y Navascués, E. L. Martín, M. G. Petr-Gotzens, J. Kolb, E. Marchetti, M. Morales-Calderón, A. Bayo, E. Artigau, M. Hartung, F. Marchis, M. Tamura, M. Sterzik, R. Köhler, V. D. Ivanov, and D. Nürnberger, "A deep look into the core of young clusters," *A&A*, vol. 504, pp. 199–209, sep 2009.

[27] "Wise - mission (nasa)."

[1] IRAS image
http://www.ipac.caltech.edu/system/projects/images/15/original/IRAS_Sq.jpg?1291155530

[2] Point Spread Function
http://zeiss-campus.magnet.fsu.edu/articles/basics/images/psfintrofigure1.jpg

[5] 3.6 meter Visible Telescope Project at Devasthal
https://www.iist.ac.in/ess/spfgi/dot-status-2014-15.pdf

[11] Planck Satellite
http://www.cosmos.esa.int/web/planck

[12] Planck Satellite image
http://www.esa.int/var/esa/storage/images/esa_multimedia/images/2007/01/planck_in_space/9880824-3-eng-GB/Planck_in_space.jpg

[13] Model of the Planck Satellite
https://upload.wikimedia.org/wikipedia/commons/e/e1/Model_of_the_Planck_Satellite_1.jpg

[15] NSSDCA/COSPAR ID: 1983-004A
http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1983-004A

[16] Skyview - Virtual Observatory (query form)
http://skyview.gsfc.nasa.gov/current/cgi/query.pl

[27] WISE Mission (NASA)
http://wise.ssl.berkeley.edu/mission.html

Appendix A

Tracking Accuracy

A.1 main.py

```
import matplotlib.pyplot as plt
import numpy as np

print (">>> Calculating centroid")
from track_star import centx, centy, exp, cd

x = centx
y = centy
t = exp
print ("...Done")
# ************************************ 30 sec acquisition data
    ****************************************
print (">>> Calculating Moving averages over 30 sec")
f1 = open(str(cd) + "/Results/centroid_sma.txt", "w")
X_sma = np.zeros(shape=((len(x) - 15), 1))
Y_sma = np.zeros(shape=((len(y) - 15), 1))
time = np.zeros(shape=((len(t) - 15), 1))
x = (x - x.mean()) * 0.05768  # field of view = 0.05768 arcsec / pixel
y = (y - y.mean()) * 0.05768
for i in range(0, len(x) - 15):
    Ix = Iy = 0
    for j in range(0, 15):
        k = i + j
        if k < len(x):
            Ix += x[k]
            Iy += y[k]
    X_sma[i] = Ix / 15
```

```
27      Y_sma[i] = Iy / 15
28      time[i] = 30 + i * 2
29      f1.write("\n %f \t %f " % (X_sma[i], Y_sma[i]))
30  f1.close()
31  time /= 3600
32  print ("...Done")
33  # ************************************ RMS on 1 minute
        ***********************************************
34  print (">>> Calculating RMS on 1 min")
35  f2 = open(str(cd) + "/Results/rms_1min.txt", "w")
36  rms_X = np.zeros(shape=((len(X_sma) - 30), 1))
37  rms_Y = np.zeros(shape=((len(Y_sma) - 30), 1))
38  rms = np.zeros(shape=((len(Y_sma) - 30), 1))
39  tym = np.zeros(shape=((len(X_sma) - 30), 1))
40
41  for i in xrange(0, len(X_sma) - 30):
42      X = X_sma[i:i + 30]
43      Y = Y_sma[i:i + 30]
44      rms_X[i] = X.std()
45      rms_Y[i] = Y.std()
46      rms[i] = np.sqrt(np.square(rms_X[i]) + np.square(rms_Y[i]))
47      tym[i] = 60 + i * 2
48      f2.write("\n %d \t %f \t %f \t %f " % (tym[i], rms_X[i], rms_Y[i],
        rms[i]))
49  f2.close()
50  rms_mean = rms.mean()
51  tym /= 3600
52  print ("...Done")
53  # ************************************ RMS on 30 minute
        ***********************************************
54  print (">>> Calculating RMS on 30 min")
55  f3 = open(str(cd) + "/Results/rms_30min.txt", "w")
56  rms_x = np.zeros(shape=((len(X_sma) - 900), 1))
57  rms_y = np.zeros(shape=((len(Y_sma) - 900), 1))
58  rms_hr = np.zeros(shape=((len(Y_sma) - 900), 1))
59  tym_hr = np.zeros(shape=((len(X_sma) - 900), 1))
60
61  for i in xrange(0, len(X_sma) - 900):
62      X = X_sma[i:i + 900]
63      Y = Y_sma[i:i + 900]
64      rms_x[i] = X.std()
65      rms_y[i] = Y.std()
66      rms_hr[i] = np.sqrt(np.square(rms_x[i]) + np.square(rms_y[i]))
67      tym_hr[i] = 1800 + i * 2
```

```
68      f3.write("\n %d \t %f \t %f \t %f " % (tym_hr[i], rms_x[i], rms_y[i
        ], rms_hr[i]))
69  f3.close()
70  rms_hr_mean = rms_hr.mean()
71  tym_hr /= 3600
72  print ("...Done")
73  # ************************************ RMS on 50 minute
        ********************************************
74  print (">>> Calculating RMS on 50 min")
75  f4 = open(str(cd) + "/Results/rms_50min.txt", "w")
76  RMS_x = np.zeros(shape=((len(X_sma) - 1500), 1))
77  RMS_y = np.zeros(shape=((len(Y_sma) - 1500), 1))
78  RMS_hr = np.zeros(shape=((len(Y_sma) - 1500), 1))
79  TYM_hr = np.zeros(shape=((len(X_sma) - 1500), 1))
80
81  for i in xrange(0, len(X_sma) - 1500):
82      X = X_sma[i:i + 1500]
83      Y = Y_sma[i:i + 1500]
84      RMS_x[i] = X.std()
85      RMS_y[i] = Y.std()
86      RMS_hr[i] = np.sqrt(np.square(rms_x[i]) + np.square(rms_y[i]))
87      TYM_hr[i] = 3000 + i * 2
88      f4.write("\n %d \t %f \t %f \t %f " % (TYM_hr[i], RMS_x[i], RMS_y[i
        ], RMS_hr[i]))
89  f4.close()
90  RMS_hr_mean = RMS_hr.mean()
91  TYM_hr /= 3600
92  print ("...Done")
93  # ************************************ Centroid Evolution Plot
        ****************************************
94  print (">>> Centroid Evolution Plot")
95  X = (X_sma - X_sma.mean())
96  Y = (Y_sma - Y_sma.mean())
97  fig1 = plt.figure()
98  plt.plot(time, X, label='X', c='blue')
99  plt.plot(time, Y, label='Y', c='magenta')
100 plt.axhline(y=0, xmin=0, xmax=1, c='red')
101 plt.legend(loc='upper right', shadow=False, fontsize=10)
102 plt.xlabel('Time (in Hr)')
103 plt.ylabel('Centroid Position (arcsec)')
104 fig1.suptitle('Closed loop tracking test - Centroid Evolution \n
        Filename : 20151130_200400_Tcam_MP_CL')
105 fig1.savefig(str(cd) + '/Results/Centroid_evolution.jpg')
106 print ("...Done")
```

```
# ********************************** Specification Analysis Plot
    **********************************
print (">>> Specification Analysis Plot")
fig2 = plt.figure()
plt.axhline(y=0.1, xmin=0, xmax=1, c='red', label='Specification') #
    Specification
plt.plot(tym, rms, c='blue', label='RMS on 1 min = ' + str(round(
    rms_mean, 3))) # RMS on 1 minute
plt.axhline(y=rms_mean, xmin=0, xmax=1, c='green') # RMS mean on 1
    minute
plt.plot(tym_hr, rms_hr, c='magenta', label='RMS on 30 min = ' + str(
    round(rms_hr_mean, 3))) # RMS on 30 minute
plt.plot(TYM_hr, RMS_hr, c='cyan', label='RMS on 50 min = ' + str(round(
    RMS_hr_mean, 3))) # RMS on 50 minute
plt.legend(loc='upper left', shadow=False, fontsize=10)
plt.xlabel('Time (in Hr)')
plt.ylabel('Tracking Error (arcsec)')
fig2.suptitle('Closed loop tracking test - Specification Analysis \n
    Filename : 20151130_200400_Tcam_MP_CL')
fig2.savefig(str(cd) + '/Results/Centroid_specification.jpg')
print ("...Done")
#
    ********************************************************************************

plt.show()
```

Listing A.1 main.py

A.2 track_star.py

```
from astropy.io import fits
from centroid import *
import numpy as np
import os


cd = "/home/samashti/Iraf/Internship/Project/Tracking_Tests/Main_Port
    /20151130_200400_Tcam_MP_CL"
fptr = open(str(cd) + "/Results/centroid_20151130_200400_Tcam_MP_CL.txt"
    , "w")


path, dirs, files = os.walk(str(cd)).next()
f_count = len(files)
```

```
centx = np.zeros(shape=(f_count, 1))
centy = np.zeros(shape=(f_count, 1))
exp = np.zeros(shape=(f_count, 1))
temp = 0
for i in range(f_count):
    filename = str(cd) + "/20151130_200400_Tcam_S" + str(i) + ".FIT"
    f = fits.open(filename)
    data = fits.getdata(filename)
    t = f[0].header['EXPTIME']
    lx, ly = data.shape
    # crop = data[lx / 2.5: -lx / 3.5, ly / 2: -ly / 5.4]
    centx[i] = centroidx(data, lx, ly)
    centy[i] = centroidy(data, lx, ly)
    temp += t
    exp[i] = temp
    fptr.write("\n %f \t %f \t %f " % (exp[i], centx[i], centy[i]))
fptr.close()
```

Listing A.2 track_star.py

A.3 centroid.py

```
import numpy


def centroidx(arr, p, q):
    xi = numpy.zeros(shape=(p, 1))
    for i in range(p):
        Ix = 0
        for j in range(q):
            Ix += arr[j, i]
        xi[i] = Ix
    I_mean = numpy.sum(xi[:]) / p
    a = 0
    b = 0
    for i in range(p):
        if xi[i] > I_mean:
            a += (xi[i] - I_mean) * i
            b += (xi[i] - I_mean)
    Xc = a / b
    return Xc  # X-coordinate of centroid


def centroidy(arr, p, q):
```

```
23      yj = numpy.zeros(shape=(q, 1))
24      for j in range(q):
25          Jy = 0
26          for i in range(p):
27              Jy += arr[j, i]
28          yj[j] = Jy
29      J_mean = numpy.sum(yj[:]) / q
30      a = 0
31      b = 0
32      for j in range(q):
33          if yj[j] > J_mean:
34              a += (yj[j] - J_mean) * (j)
35              b += (yj[j] - J_mean)
36      Yc = a / b
37      return Yc  # Y-coordinate of centroid
```

Listing A.3 centroid.py

Appendix B

Temperature Plot of molecular cloud

B.1 readme.txt

```
# ************************************************************************
# Author: Nikhil S Hubballi <nikhil.hubballi@gmail.com>                  *
# Date: July, 2016                                                       *
# Purpose: to fit modified black body function to the observed data points   *
# from Planck and 100 micron IRAS fluxes. Query and get temperature plot,    *
# optical depth plot, spectral emissivity index plot and also the relation   *
# between temperature and spectral emissivity index for the entire region    *
# of molecular cloud under inspection                                    *
# ************************************************************************


Packages required for the python program
1. astropy.io
2. lmfit
3. matplotlib
4. numpy
5. scipy
6. astropy.convolution
7. easygui
8. os

Steps to be followed to obtain the temperature map
1. Use Skyview to obtain Planck and IRAS satellite data.
```

```
url: http://skyview.gsfc.nasa.gov/current/cgi/query.pl

2. Specify the images size in both pixels and degrees such that plate scale of
   the image is 1.7 arcmin.

   Formula = Image size (degrees) * 60 / Image size (pixels) = 1.7

3. Download the data corresponding to Planck 143, 217, 353, 545, 857 GHz and
   iris 100 corresponding to IRAS.

4. Place the above images in the folder named "Data" (present in the root
   folder of the program)

5. After opening the terminal from the directory where the programs are
    located, run the main.py python file.

~/...../$ python main.py

6. The code will start depending on the proper verification FITS files
   present in the directory.

7. A user dialogue opens from where each of the corresponding FITS files
   have to be chosen by the user.

8. The code will take around 2 hours depending on the dimensions of the
   fits images and the processor capabilities.

9. After the code completion, you'll see three fits files named
   "Temp_plot.fits", "Optical_depth.fits", "Beta.fits", "data.dat",
   "temp_hist.png", "beta_hist.png".

10. For correlation plot, run the "contour_beta_temp.py" python file.
```

B.2 run.py

```
#
    ********************************************************************************
# Author: Nikhil S Hubballi <nikhil.hubballi@gmail.com>
        *
# Date: July, 2016
        *
# Purpose: to fit modified black body function to the observed data
    points    *
# from Planck and 100 micron IRAS fluxes. Query and get temperature plot
    ,    *
# optical depth plot, spectral emissivity index plot and also the
    relation    *
# between temperature and spectral emissivity index for the entire
    region    *
# of molecular cloud under inspection
        *
#
    ********************************************************************************


"""
Program to find the temperature map, spectral emissivity index map,
    optical depth map
of the molecular cloud
Inputs: 1. Planck images (143GHz, 217GHz, 353GHz, 545GHz, 857GHz) of the
     source
        2. IRAS image (Iris 100 micro meter) (fits images)
        with header files containing pixel scale ('CDELT1','CDELT2'),
    dimensions
Outputs: 1. Temperature plot (Temp) of the whole region of molecular
    cloud
         2. Optical depth plot (tau0)
         3. Spectral emissivity plot (beta)
         4. Temperature vs Beta plot (Relation)
         5. Histograms of temperature and beta
"""
from psf import *
from pypher import *
from lmfit import minimize, Parameters
import matplotlib.pyplot as plt
from astropy.io import fits
from numpy import *
import numpy as np
```

```
30 from astropy.convolution import convolve
31 import easygui
32
33 # Read the input images
    ************************************************************
34 print ('>>> Input data')
35 cd = os.getcwd()
36 planck143 = easygui.fileopenbox("143GHz image")
37 f143 = fits.open(planck143)
38 data143 = fits.getdata(planck143)
39 print ('>>>  143 GHz image received')
40
41 planck217 = easygui.fileopenbox("217GHz image")
42 f217 = fits.open(planck217)
43 data217 = fits.getdata(planck217)
44 print ('>>>  217 GHz image received')
45
46 planck353 = easygui.fileopenbox("353GHz image")
47 f353 = fits.open(planck353)
48 data353 = fits.getdata(planck353)
49 print ('>>>  353 GHz image received')
50
51 planck545 = easygui.fileopenbox("545GHz image")
52 f545 = fits.open(planck545)
53 data545 = fits.getdata(planck545)
54 print ('>>>  545 GHz image received')
55
56 planck857 = easygui.fileopenbox("857GHz image")
57 f857 = fits.open(planck857)
58 data857 = fits.getdata(planck857)
59 print ('>>>  857 GHz image received')
60
61 iras3000 = easygui.fileopenbox("IRIS100 image")
62 f3000 = fits.open(iras3000)
63 data3000 = fits.getdata(iras3000)
64 print ('>>>  3000 GHz image received')
65
66 '''Dimension of the images'''
67 dimx = int(f857[0].header['NAXIS1'])
68 dimy = int(f857[0].header['NAXIS2'])
69 print ('Image data received....')
70
71 # PSF and Kernel Generation
    ************************************************************
```

```
72 print ('>>> Generating PSF and Kernel')
73 pl_scale = float(np.abs(f857[0].header['CDELT1'])) * 60  # Pixel Scale
74 FWHM = [7.1 / pl_scale, 5.5 / pl_scale, 5.0 / pl_scale, 5.0 / pl_scale,
       5.0 / pl_scale, 2 / pl_scale]
75 fwhm_ref = np.amax(FWHM)
76 k = 'target'
77 psf_target = PointSpreadFunction(fwhm_ref, fwhm_ref, k)  # Reference PSF
       : psf143 in this case
78 psf_target = str(cd) + '/PSF/psf' + str(k) + '.fits'
79 psf = ['PSF1', 'PSF2', 'PSF3', 'PSF4', 'PSF5', 'PSF6']
80 kernel = ['K1', 'K2', 'K3', 'K4', 'K5', 'K6']
81 for i in range(len(FWHM)):
82     psf[i] = PointSpreadFunction(FWHM[i], fwhm_ref, i)  # PSF (re-
       sampled to reference psf size)
83     psf_source = str(cd) + '/PSF/psf' + str(i) + '.fits'
84     kernel[i] = pypher(psf_source, psf_target, kernel[i], pl_scale)  #
       Kernel
85 print ('Done...')
86 # Convolution of images
       ************************************************************
87 print ('>>> Convolving Images')
88 pix_143 = convolve(data143, kernel[0])
89 pix_143 *= 358.040  # Conversion from K-->MJy/sr
90 fits.writeto(str(cd) + '/Convolved/conv143.fits', pix_143, clobber=True)
91 # *******
92 pix_217 = convolve(data217, kernel[1])
93 pix_217 *= 415.465  # Conversion from K-->MJy/sr
94 fits.writeto(str(cd) + '/Convolved/conv217.fits', pix_217, clobber=True)
95 # *******
96 pix_353 = convolve(data353, kernel[2])
97 pix_353 *= 246.543  # Conversion from K-->MJy/sr
98 fits.writeto(str(cd) + '/Convolved/conv353.fits', pix_353, clobber=True)
99 # *******
100 pix_545 = convolve(data545, kernel[3])
101 fits.writeto(str(cd) + '/Convolved/conv545.fits', pix_545, clobber=True)
102 # *******
103 pix_857 = convolve(data857, kernel[4])
104 fits.writeto(str(cd) + '/Convolved/conv857.fits', pix_857, clobber=True)
105 # *******
106 pix_3000 = convolve(data3000, kernel[5])
107 fits.writeto(str(cd) + '/Convolved/conv3000.fits', pix_3000, clobber=
       True)
108 print ('Done...')
```

```
# Pixel by pixel black body curve fitting
    ***********************************************************************

print ('>>> Fitting black body curve')
'''define objective function: returns the array to be minimized'''


def fcn2min(params, x, data):
    """ model decaying sine wave, subtract data"""
    tau0 = params['tau0'].value
    beta = params['beta'].value
    Td = params['Td'].value

    model = A11 * tau0 * (xnu / 1200.0e9) ** beta * xnu ** 3 / (np.exp(
    A12 * (xnu / Td)) - 1)

    return model - yvalu


tmp = np.zeros(shape=(dimx, dimy))
beta = np.zeros(shape=(dimx, dimy))
tau0 = np.zeros(shape=(dimx, dimy))
f = open(str(cd) + '/Results/data.dat', 'w')
for x in range(dimx):
    for y in range(dimy):
        # ************************************************************
        A11 = 1.4733e-50 / 1.0e-20
        A12 = 4.8e-11
        # ************************************************************
        # data to be fitted
        xwave = np.array([100.0, 350.0, 550.0, 850.0, 1382.5, 2100.0])
        xnu = 3.0e8 / (xwave * 1.0e-6)
        yvalu = np.array([pix_3000[x][y], pix_857[x][y], pix_545[x][y],
    pix_353[x][y], pix_217[x][y], pix_143[x][y]])
        # ************************************************************
        params = Parameters()
        params.add('tau0', value=1.0e-3, min=1.0e-5)
        params.add('beta', value=2.0, min=1.0, max=2.2)
        params.add('Td', value=20.0, min=2.7, max=1000.0)
        # do fit, here with least square model
        result = minimize(fcn2min, params, args=(xwave, yvalu))
        # calculate final result
        final = yvalu + result.residual
        # write error report
```

```
149         # report_fit(result.params)
150         # ********************************************************
151         plck = 6.63e-34
152         kb = 1.38e-23
153         cl = 3.0e8
154         tmp[x][y] = result.params['Td'].value
155         tau0[x][y] = result.params['tau0'].value
156         beta[x][y] = result.params['beta'].value
157         nu0 = 1200.0e9
158         print("%f \t %f \t %f \t %f \t %f \n" % (x, y, tmp[x][y], tau0[x
    ][y], beta[x][y]))
159         f.write('{0:f} {1:f} {2:f} {3:f} {4:f} \n'.format(x, y, tmp[x][y
    ], tau0[x][y], beta[x][y]))
160 f.close()
161 print ('Done...')
162 #  Results and Plots
    ************************************************************
163 print ('>>> plotting results')
164 ''' Temperature Plot of the cloud '''
165 fig1 = plt.figure(1)
166 fig1 = plt.imshow(tmp, origin='lower')
167 fig1 = plt.colorbar()
168 fits.writeto(str(cd) + '/Results/Temp_plot.fits', tmp, clobber=True)
169
170 ''' Optical Depth map of the cloud '''
171 fig2 = plt.figure(2)
172 fig2 = plt.imshow(tau0, origin='lower')
173 fig2 = plt.colorbar()
174 fits.writeto(str(cd) + '/Results/Optical_depth.fits', tau0, clobber=True
    )
175
176 ''' Spectral Emissivity index map of the cloud '''
177 fig3 = plt.figure(3)
178 fig3 = plt.imshow(beta, origin='lower')
179 fig3 = plt.colorbar()
180 fits.writeto(str(cd) + '/Results/Beta.fits', beta, clobber=True)
181
182 ''' Relation between Temperature and spectral emissivity index '''
183 fig4 = plt.figure(4)
184 fig4 = plt.scatter(tmp, beta, s=0.01, edgecolor='red')
185 fig4 = plt.xlim(12, 22)
186 fig4 = plt.ylim(1.2, 2.2)
187 fig4 = plt.xlabel(r'$Temperature$')
188 fig4 = plt.ylabel(r'$Beta$, Spectral Emissivity Index')
```

```
189 fig4 = plt.grid(True)
190 fig4 = plt.savefig(str(cd) + '/Results/Temp_beta.png')
191
192 ''' Histogram of temperature values '''
193 hist1 = plt.figure(5)
194 data = loadtxt(str(cd) + "/Results/data.dat")
195 x = data[:, 2]
196 bins = np.linspace(10, 25, 1000)
197 y, binEdges = np.histogram(x, bins)  # alpha=0.5
198 bincenters = 0.5 * (binEdges[1:] + binEdges[:-1])
199 hist1 = plt.plot(bincenters, y, '-')
200 hist1 = plt.xlabel(r'$Temperature$')
201 hist1 = plt.ylabel('Number of Pixels')
202 hist1 = plt.savefig(str(cd) + '/Results/Temp_hist.png')
203
204 ''' Histogram of beta values '''
205 hist2 = plt.figure(6)
206 x = data[:, 4]
207 bins = np.linspace(1.0, 2.1, 500)
208 y, binEdges = np.histogram(x, bins)  # alpha=0.5
209 bincenters = 0.5 * (binEdges[1:] + binEdges[:-1])
210 hist2 = plt.plot(bincenters, y, '-')
211 hist2 = plt.xlabel(r'$Beta$')
212 hist2 = plt.ylabel('Number of Pixels')
213 hist2 = plt.savefig(str(cd) + '/Results/Beta_hist.png')
214 print ('Done...')
215 #
    **********************************************************************

216 plt.show()
```

Listing B.1 run.py

B.3 psf.py

```
1 from astropy.io import fits
2 from astropy.convolution import Gaussian2DKernel as G2Dk
3 import scipy.ndimage.interpolation as congrid
4 import os
5
6 cd = os.getcwd()
7
8
9 def PointSpreadFunction(sigma_source, sigma_target, ref):
```

```
    """
        Generate the PSF with given parameters

        Parameters
        ----------
        sigma_source : `float`
            source psf standard deviation
        sigma_target : 'float'
            target psf standard deviation
        ref : int
            File number

        Returns
        -------
        output : `numpy.ndarray`
            PSF data array
    """
    psf_source = G2Dk(stddev=sigma_source/2.35482)
    psf_target = G2Dk(stddev=sigma_target/2.35482)
    fac = float(psf_target.shape[0]) / float(psf_source.shape[0])
    psf_resized = congrid.zoom(psf_source, fac, order=3)
    hdu = fits.PrimaryHDU(psf_resized)
    hdulist = fits.HDUList([hdu])
    hdulist.writeto(str(cd)+'/PSF/psf'+str(ref)+'.fits', clobber=True)
    return psf_resized
```

Listing B.2 psf.py

B.4 pypher.py

```
from __future__ import absolute_import, print_function, division
import os
import numpy as np
from scipy.ndimage import rotate, zoom
from astropy.io import fits

cd = os.getcwd()
#######
# MAIN
#######


def pypher(psf_source, psf_target, output, pix_scale):

```

```
    angle_source = 0
    angle_target = 0

    kernel_basename, _ = os.path.splitext(output)
    kernel_fits = str(cd)+'/Kernel/'+kernel_basename + '.fits'

    # Load images (NaNs are set to 0)
    psf_source = fits.getdata(psf_source)
    psf_target = fits.getdata(psf_target)

    # Set NaNs to 0.0
    psf_source = np.nan_to_num(psf_source)
    psf_target = np.nan_to_num(psf_target)

    # Retrieve the pixel scale of each image
    pixscale_source = pixscale_target = pix_scale

    # Rotate images (if necessary)
    if angle_source != 0.0:
        psf_source = imrotate(psf_source, angle_source)
    if angle_target != 0.0:
        psf_target = imrotate(psf_target, angle_target)

    # Normalize the PSFs
    psf_source /= psf_source.sum()
    psf_target /= psf_target.sum()

    # Resample high resolution image to the low one
    if pixscale_source != pixscale_target:
        psf_source = imresample(psf_source, pixscale_source,
pixscale_target)

    # check the new size of the source vs. the target
    if psf_source.shape > psf_target.shape:
        psf_source = trim(psf_source, psf_target.shape)
    else:
        psf_source = zero_pad(psf_source, psf_target.shape, position='
center')
    reg_fact = 1E-4
    kernel, _ = homogenization_kernel(psf_target, psf_source, reg_fact)

    # Write kernel to FITS file
    fits.writeto(kernel_fits, data=kernel, clobber=True)
    return kernel
```

```
57
58
59 ################
60 # IMAGE METHODS
61 ################
62
63
64 def imrotate(image, angle, interp_order=1):
65     """
66     Rotate an image from North to East given an angle in degrees
67
68     Parameters
69     ----------
70     image : `numpy.ndarray`
71         Input data array
72     angle : float
73         Angle in degrees
74     interp_order : int, optional
75         Spline interpolation order [0, 5] (default 1: linear)
76
77     Returns
78     -------
79     output : `numpy.ndarray`
80         Rotated data array
81
82     """
83     return rotate(image, -1.0 * angle, order=interp_order, reshape=False
    , prefilter=False)
84
85
86 def imresample(image, source_pscale, target_pscale, interp_order=3):
87     """
88     Resample data array from one pixel scale to another
89
90     The resampling ensures the parity of the image is conserved
91     to preserve the centering.
92
93     Parameters
94     ----------
95     image : `numpy.ndarray`
96         Input data array
97     source_pscale : float
98         Pixel scale of ``image`` in arcseconds
99     target_pscale : float
```

```
        Pixel scale of output array in arcseconds
    interp_order : int, optional
        Spline interpolation order [0, 5] (default 1: linear)

    Returns
    _______
    output : `numpy.ndarray`
        Resampled data array

    """
    old_size = image.shape[0]
    new_size_raw = old_size * source_pscale / target_pscale
    new_size = int(np.ceil(new_size_raw))

    if new_size > 10000:
        raise MemoryError("The resampling will yield a too large image.
    "
                          "Please resize the input PSF image.")

    # Chech for parity
    if not (old_size - new_size) % 2:
        new_size += 1

    ratio = new_size / old_size

    return zoom(image, ratio, order=interp_order) / ratio**2


def trim(image, shape):
    """
    Trim image to a given shape

    Parameters
    __________
    image: 2D `numpy.ndarray`
        Input image
    shape: tuple of int
        Desired output shape of the image

    Returns
    _______
    new_image: 2D `numpy.ndarray`
        Input image trimmed

```

```
143     """
144     shape = np.asarray(shape, dtype=int)
145     imshape = np.asarray(image.shape, dtype=int)
146 
147     if np.alltrue(imshape == shape):
148         return image
149 
150     if np.any(shape <= 0):
151         raise ValueError("TRIM: null or negative shape given")
152 
153     dshape = imshape - shape
154     if np.any(dshape < 0):
155         raise ValueError("TRIM: target size bigger than source one")
156 
157     if np.any(dshape % 2 != 0):
158         raise ValueError("TRIM: source and target shapes "
159                          "have different parity")
160 
161     idx, idy = np.indices(shape)
162     offx, offy = dshape // 2
163 
164     return image[idx + offx, idy + offy]
165 
166 
167 def zero_pad(image, shape, position='corner'):
168     """
169     Extends image to a certain size with zeros
170 
171     Parameters
172     ----------
173     image: real 2d `numpy.ndarray`
174         Input image
175     shape: tuple of int
176         Desired output shape of the image
177     position : str, optional
178         The position of the input image in the output one:
179             * 'corner'
180                 top-left corner (default)
181             * 'center'
182                 centered
183 
184     Returns
185     -------
186     padded_img: real `numpy.ndarray`
```

```
        The zero-padded image

    """
    shape = np.asarray(shape, dtype=int)
    imshape = np.asarray(image.shape, dtype=int)

    if np.alltrue(imshape == shape):
        return image

    if np.any(shape <= 0):
        raise ValueError("ZERO_PAD: null or negative shape given")

    dshape = shape - imshape
    if np.any(dshape < 0):
        raise ValueError("ZERO_PAD: target size smaller than source one")

    pad_img = np.zeros(shape, dtype=image.dtype)

    idx, idy = np.indices(imshape)

    if position == 'center':
        if np.any(dshape % 2 != 0):
            raise ValueError("ZERO_PAD: source and target shapes have different parity.")
        offx, offy = dshape // 2
    else:
        offx, offy = (0, 0)

    pad_img[idx + offx, idy + offy] = image

    return pad_img


##########
# FOURIER
##########


def udft2(image):
    """Unitary fft2"""
    norm = np.sqrt(image.size)
    return np.fft.fft2(image) / norm

```

```
229
230 def uidft2(image):
231     """Unitary ifft2"""
232     norm = np.sqrt(image.size)
233     return np.fft.ifft2(image) * norm
234
235
236 def psf2otf(psf, shape):
237     """
238     Convert point-spread function to optical transfer function.
239
240     Compute the Fast Fourier Transform (FFT) of the point-spread
241     function (PSF) array and creates the optical transfer function (OTF)
242     array that is not influenced by the PSF off-centering.
243     By default, the OTF array is the same size as the PSF array.
244
245     To ensure that the OTF is not altered due to PSF off-centering,
    PSF2OTF
246     post-pads the PSF array (down or to the right) with zeros to match
247     dimensions specified in OUTSIZE, then circularly shifts the values
    of
248     the PSF array up (or to the left) until the central pixel reaches
    (1,1)
249     position.
250
251     Parameters
252     ----------
253     psf : `numpy.ndarray`
254         PSF array
255     shape : int
256         Output shape of the OTF array
257
258     Returns
259     -------
260     otf : `numpy.ndarray`
261         OTF array
262
263     Notes
264     -----
265     Adapted from MATLAB psf2otf function
266
267     """
268     if np.all(psf == 0):
269         return np.zeros_like(psf)
```

```
270
271     inshape = psf.shape
272     # Pad the PSF to outsize
273     psf = zero_pad(psf, shape, position='corner')
274
275     # Circularly shift OTF so that the 'center' of the PSF is
276     # [0,0] element of the array
277     for axis, axis_size in enumerate(inshape):
278         psf = np.roll(psf, -int(axis_size / 2), axis=axis)
279
280     # Compute the OTF
281     otf = np.fft.fft2(psf)
282
283     # Estimate the rough number of operations involved in the FFT
284     # and discard the PSF imaginary part if within roundoff error
285     # roundoff error  = machine epsilon = sys.float_info.epsilon
286     # or np.finfo().eps
287     n_ops = np.sum(psf.size * np.log2(psf.shape))
288     otf = np.real_if_close(otf, tol=n_ops)
289
290     return otf
291
292
293 ################
294 # DECONVOLUTION
295 ################
296
297 LAPLACIAN = np.array([[ 0, -1,  0],
298                       [-1,  4, -1],
299                       [ 0, -1,  0]])
300
301
302 def deconv_wiener(psf, reg_fact):
303     r"""
304     Create a Wiener filter using a PSF image
305
306     The signal is $\ell_2$ penalized by a 2D Laplacian operator that
307     serves as a high-pass filter for the regularization process.
308     The key to the process is to use optical transfer functions (OTF)
309     instead of simple Fourier transform, since it ensures the phase
310     of the psf is adequately placed.
311
312     Parameters
313     ----------
```

```
314     psf: `numpy.ndarray`
315         PSF array
316     reg_fact: float
317         Regularisation parameter for the Wiener filter
318 
319     Returns
320     -------
321     wiener: complex `numpy.ndarray`
322         Fourier space Wiener filter
323 
324     """
325     # Optical transfer functions
326     trans_func = psf2otf(psf, psf.shape)
327     reg_op = psf2otf(LAPLACIAN, psf.shape)
328 
329     wiener = np.conj(trans_func) / (np.abs(trans_func)**2 +
330                                     reg_fact * np.abs(reg_op)**2)
331 
332     return wiener
333 
334 
335 def homogenization_kernel(psf_target, psf_source, reg_fact=1e-4, clip=
    True):
336     """
337     Compute the homogenization kernel to match two PSFs
338 
339     The deconvolution step is done using a Wiener filter with $\ell_2$
340     penalization.
341     The output is given both in Fourier and in the image domain to serve
342     different purposes.
343 
344     Parameters
345     ----------
346     psf_target: `numpy.ndarray`
347         2D array
348     psf_source: `numpy.ndarray`
349         2D array
350     reg_fact: float, optional
351         Regularisation parameter for the Wiener filter
352     clip: bool, optional
353         If `True`, enforces the non-amplification of the noise
354         (default `True`)
355 
356     Returns
```

```
    -------
    kernel_image: `numpy.ndarray`
        2D deconvolved image
    kernel_fourier: `numpy.ndarray`
        2D discrete Fourier transform of deconvolved image

    """
    wiener = deconv_wiener(psf_source, reg_fact)

    kernel_fourier = wiener * udft2(psf_target)
    kernel_image = np.real(uidft2(kernel_fourier))

    if clip:
        kernel_image.clip(-1, 1)

    return kernel_image, kernel_fourier


New BSD License

Copyright (c) 2015 IAS / CNRS / Univ. Paris-Sud
All rights reserved.
```

Listing B.3 pypher.py

B.5 pixbypix.py

```
#
    ********************************************************************************
# Author: Nikhil S Hubballi
        *
# Date: July, 2016
        *
# Purpose: to fit modified black body function to the observed data
    points    *
# from Planck and 100 micron IRAS fluxes. Query and get results for
        *
# for individual pixels.
        *
#
    ********************************************************************************

"""
```

```
Program to find the black body curve fit of the molecular cloud for any position in the image
Inputs: 1. Planck images (143GHz, 217GHz, 353GHz, 545GHz, 857GHz) of the source
        2. IRAS image (Iris 100 micro meter) (fits images)
        with header files containing pixel scale ('CDELT1','CDELT2'), dimensions
Output: Black body curve fit plot for the position in the molecular cloud with
        temperature and spectral emissivity index(beta) values
"""
from psf import *
from pypher import *
from lmfit import minimize, Parameters, report_fit
import matplotlib.pyplot as plt
from astropy.io import fits
from numpy import *
import numpy as np
from astropy.convolution import convolve
import easygui


# Read the input images ****************************************************
print ('>>> Input data')
cd = os.getcwd()

planck143 = easygui.fileopenbox("143 Ghz image")
f143 = fits.open(planck143)
data143 = fits.getdata(planck143)
print ('>>>  143 GHz image received')

planck217 = easygui.fileopenbox("217 Ghz image")
f217 = fits.open(planck217)
data217 = fits.getdata(planck217)
print ('>>>  217 GHz image received')

planck353 = easygui.fileopenbox("353 Ghz image")
f353 = fits.open(planck353)
data353 = fits.getdata(planck353)
print ('>>>  353 GHz image received')

planck545 = easygui.fileopenbox("545 Ghz image")
f545 = fits.open(planck545)
```

```
48 data545 = fits.getdata(planck545)
49 print ('>>>  545 GHz image received')
50
51 planck857 = easygui.fileopenbox("857 Ghz image")
52 f857 = fits.open(planck857)
53 data857 = fits.getdata(planck857)
54 print ('>>>  857 GHz image received')
55
56 iras3000 = easygui.fileopenbox("IRIS100 image")
57 f3000 = fits.open(iras3000)
58 data3000 = fits.getdata(iras3000)
59 print ('>>>  3000 GHz image received')
60
61 '''Dimension of the images'''
62 dimx = float(f857[0].header['NAXIS1'])
63 dimy = float(f857[0].header['NAXIS2'])
64 print ('Image data received....')
65
66 # PSF and Kernel Generation
    ********************************************
67 print ('>>> Generating PSF and Kernel')
68 pl_scale = float(np.abs(f857[0].header['CDELT1'])) * 60  # Pixel Scale
69 FWHM = [7.1 / pl_scale, 5.5 / pl_scale, 5.0 / pl_scale, 5.0 / pl_scale,
    5.0 / pl_scale, 2.0 / pl_scale]
70 fwhm_ref = np.amax(FWHM)
71 k = 'target'
72 psf_target = PointSpreadFunction(fwhm_ref, fwhm_ref, k)  # Reference PSF
    : psf143 in this case
73 psf_target = str(cd)+'/PSF/psf'+str(k)+'.fits'
74 psf = ['PSF1', 'PSF2', 'PSF3', 'PSF4', 'PSF5', 'PSF6']
75 kernel = ['K1', 'K2', 'K3', 'K4', 'K5', 'K6']
76 for i in range(len(FWHM)):
77     psf[i] = PointSpreadFunction(FWHM[i], fwhm_ref, i+1)  # PSF (re-
    sampled to reference psf size)
78     psf_source = str(cd)+'/PSF/psf'+str(i)+'.fits'
79     kernel[i] = pypher(psf_source, psf_target, kernel[i], pl_scale)  #
    Kernel
80 print ('Done...')
81
82 # Convolution of images
    ***************************************************
83 print ('>>> Convolving Images')
84 pix_143 = convolve(data143, kernel[0])
85 pix_143 *= 358.040  # Conversion from K-->MJy/sr
```

```
fits.writeto(str(cd)+'/Convolved/conv143.fits', pix_143, clobber=True)
# *******
pix_217 = convolve(data217, kernel[1])
pix_217 *= 415.465  # Conversion from K-->MJy/sr
fits.writeto(str(cd)+'/Convolved/conv217.fits', pix_217, clobber=True)
# *******
pix_353 = convolve(data353, kernel[2])
pix_353 *= 246.543  # Conversion from K-->MJy/sr
fits.writeto(str(cd)+'/Convolved/conv353.fits', pix_353, clobber=True)
# *******
pix_545 = convolve(data545, kernel[3])
fits.writeto(str(cd)+'/Convolved/conv545.fits', pix_545, clobber=True)
# *******
pix_857 = convolve(data857, kernel[4])
fits.writeto(str(cd)+'/Convolved/conv857.fits', pix_857, clobber=True)
# *******
pix_3000 = convolve(data3000, kernel[5])
fits.writeto(str(cd)+'/Convolved/conv3000.fits', pix_3000, clobber=True)
print ('Done...')

# Pixel by pixel black body curve fitting
    ***********************************
# ******INPUT******
inp = raw_input('Try? (y/n) :')

while str(inp) == 'y':
    x = input("Provide x co-ord: ")
    y = input("\nProvide y co-ord: ")
    print ('>>> Fitting black body curve')
    # ************************************************************
    A11 = 1.4733e-50 / 1.0e-20
    A12 = 4.8e-11
    # ************************************************************
    # data to be fitted
    xwave = np.array([100.0, 350.0, 550.0, 850.0, 1382.5, 2100.0])
    xnu = 3.0e8 / (xwave * 1.0e-6)
    yvalu = np.array([pix_3000[y][x], pix_857[y][x], pix_545[y][x],
    pix_353[y][x], pix_217[y][x], pix_143[y][x]])

    # ************************************************************
    '''define objective function: returns the array to be minimized'''

    def fcn2min(params, x, data):
        """ model decaying sine wave, subtract data"""
```

```
128         tau0 = params['tau0'].value
129         beta = params['beta'].value
130         Td = params['Td'].value
131 
132         model = A11 * tau0 * (xnu / 1200.0e9) ** beta * xnu ** 3 / (np.
    exp(A12 * (xnu / Td)) - 1)
133         return model - yvalu
134 
135 
136     params = Parameters()
137     params.add('tau0', value=1.0e-3, min=1.0e-5)
138     params.add('beta', value=2.0, min=1.0, max=2.2)
139     params.add('Td', value=20.0, min=2.7, max=1000.0)
140 
141     # do fit, here with least square model
142     result = minimize(fcn2min, params, args=(xwave, yvalu))
143     # calculate final result
144     final = yvalu + result.residual
145     # write error report
146     report_fit(result.params)
147     # ***************************************************
148     plck = 6.63e-34
149     kb = 1.38e-23
150     cl = 3.0e8
151     Temp = result.params['Td'].value
152     tau0 = result.params['tau0'].value
153     beta = result.params['beta'].value
154     nu0 = 1200.0e9
155 
156     # ***************************************************
157     xw = empty(3000)
158     for i in range(1, 3000):
159         xw[i] = i
160     f1 = open(str(cd)+'/Results/Inu_nuTFIT.dat', 'w')
161     for j in range(1, 3000):
162         nu = cl / (xw[j] * 1.0e-6)
163         A11 = (2 * plck * nu * nu * nu) / (cl * cl)
164         A12 = (plck * nu) / (kb * Temp)
165         A13 = 1 / (exp(A12) - 1)
166         A14 = tau0 * pow((nu / nu0), beta)
167         Inu = A11 * A13 * A14 * 1.0e+20
168         f1.write('{0:f} {1:f} {2:.30f}\n'.format(nu, xw[j], Inu))
169     f1.close()
```

```
170     nuFITa, wlFITa, InuFITa = loadtxt(str(cd)+'/Results/Inu_nuTFIT.dat',
        unpack=True)
171     print ('Model fitted...')
172     print ('Plotting the blackbody fit...')
173     # ************************************************
174     plt.figure(figsize=(10, 7))
175     # Black Body fit (continuous)
176     plt.plot(wlFITa, InuFITa, 'b-', label="Modified BB Fit")
177     # Black body fit values (data points)
178     plt.plot(xwave, final, 'rs', markersize=10, markerfacecolor='white',
        label="Fit Values")
179     # Pixel values
180     plt.plot(xwave, yvalu, 'ro', label="Pixel Value")
181     # plot parameters
182     plt.xscale('log'), plt.yscale('log')
183     plt.xlim(30.0, 5000.0), plt.ylim(0.01, 1000.0)
184     plt.text(40, 300, 'T = '+str(round(Temp, 3))+'K , Beta = '+str(round
        (beta, 3)), fontdict=None)
185     plt.xlabel('Wavelength (microns)'), plt.ylabel('Intensity (MJy/sr)')
186     plt.legend(loc='upper right'), plt.legend(numpoints=1)
187     plt.savefig(str(cd)+'/Results/'+str(T)+'.png')
188     print ('Done...')
189     # Plot
        ************************************************************
190     plt.show()
191     inp = raw_input('Try another? (y/n) :')
```

Listing B.4 pixbypix.py

B.6 coversion.py

```
1  from astropy.io import fits
2  from numpy import *
3  import easygui
4  import os
5
6  # Read the input images ********************************************
7  cd = os.getcwd()
8  planck100 = easygui.fileopenbox("100GHz image")
9  f100 = fits.open(planck100)
10 data100 = fits.getdata(planck100)
11
12 planck143 = easygui.fileopenbox("143GHz image")
13 f143 = fits.open(planck143)
```

```
data143 = fits.getdata(planck143)

planck217 = easygui.fileopenbox("217GHz image")
f217 = fits.open(planck217)
data217 = fits.getdata(planck217)

planck353 = easygui.fileopenbox("353GHz image")
f353 = fits.open(planck353)
data353 = fits.getdata(planck353)

planck545 = easygui.fileopenbox("545GHz image")
f545 = fits.open(planck545)
data545 = fits.getdata(planck545)

planck857 = easygui.fileopenbox("857GHz image")
f857 = fits.open(planck857)
data857 = fits.getdata(planck857)

iras3000 = easygui.fileopenbox("IRAS100 image")
f3000 = fits.open(iras3000)
data3000 = fits.getdata(iras3000)

data100 *= 218.200
data143 *= 358.040
data217 *= 415.465
data353 *= 246.543
fits.writeto('convrt100.fits', data100, clobber=True)
fits.writeto('convrt143.fits', data143, clobber=True)
fits.writeto('convrt217.fits', data217, clobber=True)
fits.writeto('convrt353.fits', data353, clobber=True)
fits.writeto('convrt545.fits', data545, clobber=True)
fits.writeto('convrt857.fits', data857, clobber=True)
fits.writeto('convrt3000.fits', data3000, clobber=True)
```

Listing B.5 conversion.py

B.7 contour_beta_temp.py

```
import matplotlib.pyplot as plt
from numpy import loadtxt
import numpy as np
import numpy
import matplotlib.pyplot as plt
from astropy.io import fits
```

```
#def plotbet_temp():
from scipy.stats import gaussian_kde
data = loadtxt("Results/data.dat")
temp = data[:,2]
beta = data[:,4]
x=temp
y=beta
xy = np.vstack([x,y])
z = gaussian_kde(xy)(xy)

fig, ax = plt.subplots()
plt.scatter(x, y, c=z, s=1, edgecolor='')
plt.xlim(12, 22)
plt.ylim(1.2,2.2)
plt.xlabel(r'$Temperature$')
plt.ylabel(r'$Beta$')
plt.grid(True)
plt.savefig('Results/Beta_Temp_corr.png')
plt.show()
```

Listing B.6 contour_beta_temp.py