Creating a Normalized Model of Noise in the Universe Based on Position Using *Herschel* PACS Data[*]

Verónica Cruz

August 1, 2020

Abstract

The *Herschel* PACS instrument captured photometric and spectrometric data from 2009 to 2013. However, background and structure noise are still an issue for many other satellite telescopes. This research consists of the creation of three maps of noise for RMS noise, SNR noise, and structure noise using the metadata from the *Herschel* PACS Point Source Catalogue at 70μm using Python's various libraries including Matplotlib and SciPy. Heat maps were created at first to visualize the data. Then, the empty space in the heat maps was interpolated to calculate noise in a given position in space. Then, interpolation was used via Python to estimate the noise values where no data was captured. Most of the data and most of the noise was found in the galactic plane, which is to be expected. These models could be used to find strange fluctuations in noise which could lead to new observations. However, the scope of this research is limited due to hardware and temporal constraints.

[*] *Herschel* is an ESA space observatory with science instruments provided by European-led Principal Investigator consortia and with important participation from NASA.

1 Introduction

The *Herschel* Space Observatory was an optical satellite telescope that detected far-infrared and sub-millimeter wavelengths [1]. *Herschel* consisted of 3 instruments: HIFI (Heterodyne Instrument for the Far Infrared), SPIRE (Spectral and Photometric Imaging Receiver), and PACS (Photodetector Array Camera and Spectrometer) [2]. Its mission was to use its detection of these wavelengths to examine how galaxies and stars have evolved since the birth of our universe and to determine the chemical compositions of the atmospheres and surfaces of various celestial objects, including comets, planets, and moons [1]. The *Herschel* telescope, which was active from 2009 to 2013, has since been decommissioned [2]. However, the data captured from this telescope and the metadata associated with it is still available in the European Space Agency archives. It is from these archives that the data was collected to create a map of signal noise.

For the purposes of this paper, I will be focusing on the metadata from the PACS instrument at 70 microns as opposed to all three color bands at 70, 100, and 160 microns [3]. Data was obtained through the ESA website in a CSV file titled "*Herschel*/PACS Point Source Catalogue @ 70μm" that included 108,319 entries [3]. The metadata included the target's name, the wavelength band (of which all data points were blue because they were all 70μm), right ascension and its error, declination and its error, flux, the signal to noise ratio, statistically derived noise (snrnoise), the ratio of flux and RMS (stn), RMS noise, structure noise (strn), flux ratio, $FWHM_x$ and $FWHM_y$, elongation flag, edge flag, blend flag, the warm attitude, the observation ID, and solar system map flag. However, this research will only focus on right ascension, declination, RMS noise, SNR noise, and structure noise [4].

Right ascension is comparable to latitude on Earth. Similarly, Declination is analogous to longitude on Earth. They both follow the alignment of Earth's equator on January 1, 2000 at 12:00 terrestrial time, more well known as J2000 [5]. Also, they do not depict the distance of celestial bodies, only their location relative to the Earth sky (Fig. 1). RMS noise is the average amount of noise in millijanskys (mJ) calculated from nearby objects whose own emissions of electromagnetic energy may interfere with the data being collected [4]. This is unlike SNR noise which is statistically determined via the signal-to-noise ratio where the signal refers to the flux of the target celestial object; however, the unit is also millijanskys (mJ) [4]. The structure noise refers to the amount of error, measured in megajanskys per steradian ($MJ \times sr^{-1}$), caused by the telescope itself [4].

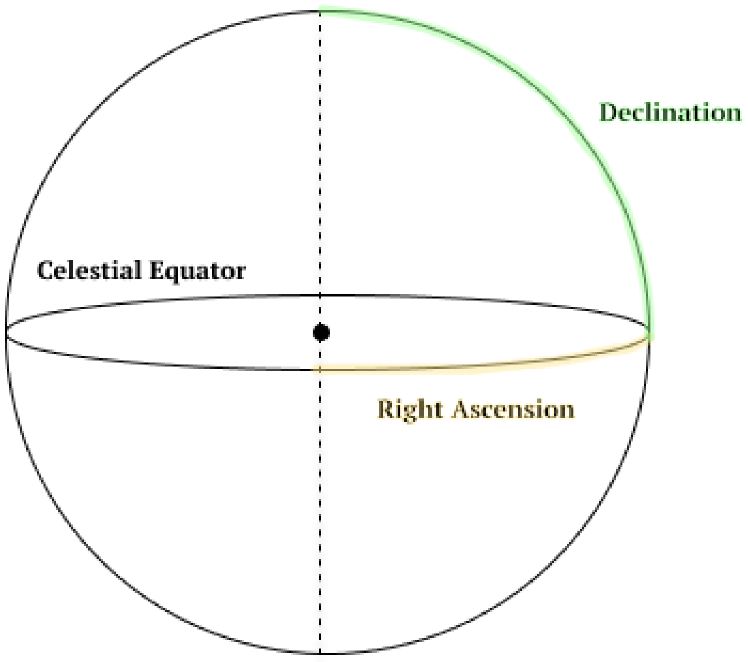

Figure 1: A diagram depicting the equatorial coordinate system. This includes right ascension and declination.

The interpolation portion of this research focuses on three, main methods of interpolation: nearest neighbor, linear, and cubic. Nearest neighbor interpolation occurs when the point being interpolated takes the value of the nearest point without factoring in other points [6]. Linear interpolation uses linear polynomials to interpolate between known data points [6]. Similarly, cubic interpolation uses third-degree polynomials to draw new data points where no data was recorded [6].

In the future, a normalized map of noise in the universe could be used to find strange fluctuations in noise that are beyond what is expected. Finding these fluctuations could lead to the observation of new, distant stars or other celestial objects, or even new celestial phenomena. Due to the novelty and constraints of this research, my results would need to be developed more but the end results could have monumental applications.

2 Literature Review

Most of the research done on the signal noise of telescopes has focused on the technical side and reducing structure noise. This is the first time that this data is being analyzed this way. Therefore, there is limited literature regarding this topic. However, there has been some research similar to that of this project. In 2012, Aniano et al. used noise measures from the *Spitzer* and *Herschel* telescopes to map out interstellar dust in the galaxies NGC 628 and NGC 6946 [7]. Additionally, in 2017, Rosen et al. mapped out signals and noise based on azimuth and elevation using contour trees and data cubes [8]. There has not been much other research done in this realm of observational astronomy.

3 Purpose

1. Create heat maps of 3 kinds of noise (rms, snrnoise, strn)
2. Use Python to interpolate noise where no data was captured
3. Create a 3D model of interpolated noise to visualize variations in data
4. Create a function that accepts floats to determine noise given position

4 Methods

Throughout this research, the Python language was used via the free version of Google Colaboratory [9]. The specific libraries imported for this research were pandas, NumPy, Matplotlib's PyPlot and mplot3d libraries, and Scipy's interpolate library.

The first objective was to create heat maps to visualize where the given data lies and what the noise values are in a given position. (Fig. 2). Heat maps represent three dimensions of data with the first two dimensions being the x and y values and the third dimension being represented using color values. The *scatter* function from the Matplotlib library of Python was used, with c equaling the respective measure of noise. For clarity, a color bar was also included. All color bars had a limit of 95 as this seemed to be the limit for outliers for all three noise measures. The outliers seemed to originate from areas that were dusty and very bright, like nebulae. As is the standard for graphing right ascension, the x-axis was inverted. The heat maps were later used to pick a method for interpolation (nearest, linear, or cubic).

The next objective—the main part of the project—was to create interpolated map of the three measures of noise which could in turn be used as a function to find noise in a given position. First, each method was tested and compared to the original, non-interpolated heat maps. The interpolated maps were created using the *griddata* function from SciPy and were shown using the *imshow* function from Matplotlib. The function used accepted three different types of interpolation: nearest, linear, and cubic [6]. The *griddata's* extent must be specified to be from 0 to 360 for x and -90 to 90 for the y or the declination will not display correctly. The xi values within *griddata* were created using NumPy's *mgrid* function with the x values ranging from 0 to 360 with 360 units and the y values ranging from -90 to 90 with 180 total units. After comparing the true, observed values of noise to the interpolated maps, it was decided that nearest neighbors would be the best method of interpolation; the interpolated values of noise seemed to match what one would approximate after analyzing the original heat map.

Finally, to better visualize the 3D grid, three-dimensional versions of the interpolated heat maps were created to show the full extent of the variations of noise and to visualize the gradient descent occurring with the *griddata* function.

To create the actual three-dimensional figure itself, the function *mplot3d.Axes3D (plt.figure())* was utilized. This was set to variable *ax* which was then added to the function *ax.plot_trisurf* with x, y, and z being the three one-dimensional arrays specified as parameters. X was a list of right ascension values in integers, y was a list of declination values in integers, and z was a list of the *griddata* values for the respective x and y indices Then, once again, a color bar was added to the figure and the x-axis was inverted according to convention. For visualization purposes, the natural log of variable z was taken due to the high variations in values and multiple outliers.

Limited hardware and temporal resources limited the extent of research to only the metadata of the *Herschel* PACS telescope at $70\mu m$. Only 13 GB of RAM were available for the duration of this research, which caused the prefered functions to crash the program entirely. Therefore, sub-optimal functions that did not directly address the research intent had to be used. For example, the *griddata* function created earlier could be used to find noise given an integer right ascension and declination. However, the function cannot be used to calculate floats as it only accepts integer indices. Additionally, there were only 6 weeks given for the development of this research, so the results may be incomplete. I encourage other astronomers to continue my research and add onto it.

5 Results and Discussion

The results for the original heat maps seemed to match what one would expect (Fig. 2). The heat maps depict the three-dimensional sky on a two-dimensional surface. The visible curve in the graph is the galactic plane. However, this was not drawn in; it is a collection of data points concentrated into a curved shape. Each point on the map is a data point from the Herschel PACS Point Source Catalogue from which the data was taken. The color corresponds to the amount of noise in the respective measure and units, with more yellow values corresponding to more noisy areas. Most of the data collected seemed to be from within the Milky Way which is understandable due to our own location within this galaxy. The noisiest areas on the map correspond to nebulae and overall bright and dusty areas in space.

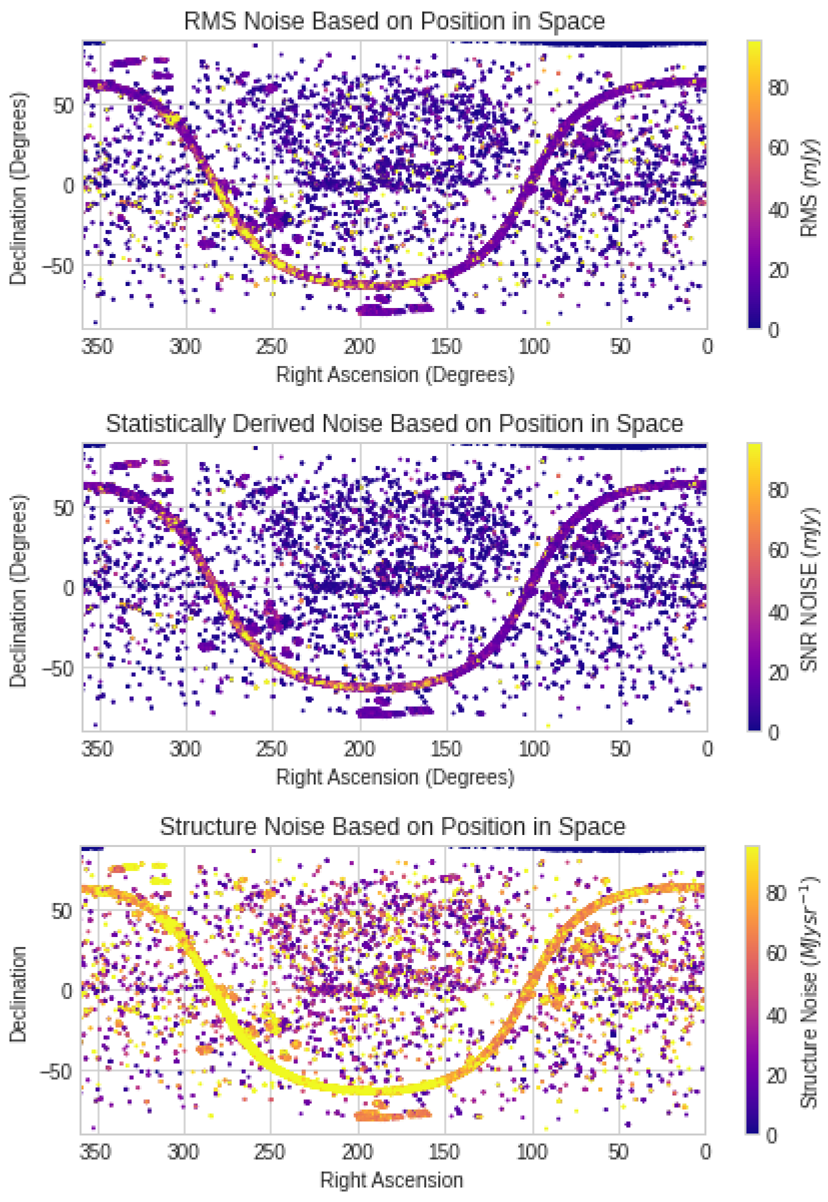

Figure 2: The three heat maps created for the first objective of this research, the first being RMS noise, the second being SNR noise, and the third being Structure Noise (STRN). Although the colorbar only reaches 95, the maxima of these noise measures can reach as high as about $4.5 \times 10^5 mJy$ for RMS noise, almost $6 \times 10^6 mJy$ for SNR Noise, and more than $1 \times 10^6 MJy \times sr^{-1}$ for structure noise.

The interpolated maps displayed some surprising results. Areas with no RMS data captured usually defaulted to have a noise measure less than 30 mJy (Fig. 3). Areas where no SNR noise was calculated were interpolated to have less than 20 mJy (Fig. 4). Structure noise, on the other hand, had a lot more variation regardless of position. The large amount of structure noise in the galactic plane was unexpected. One would think that structure noise would be independent of the target's position. This unexpected result will be investigated further in the future.

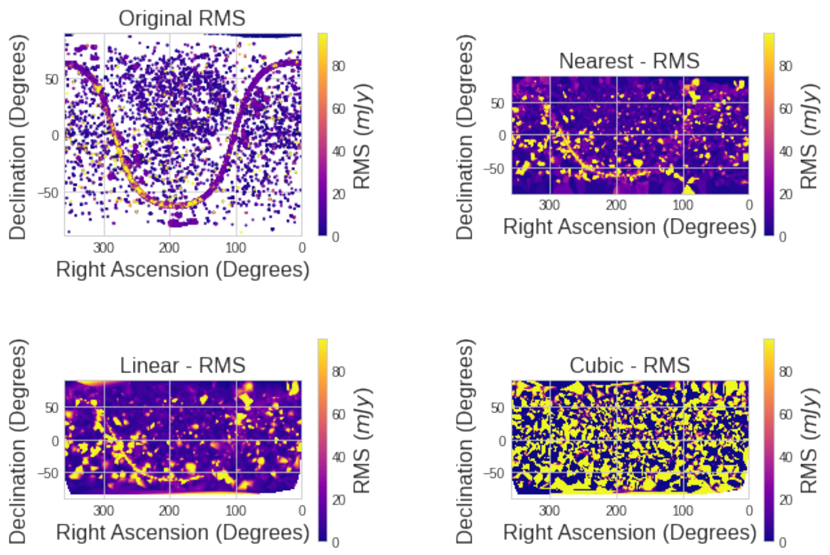

Figure 3: The different methods of interpolation tested for RMS noise. The various methods were compared to the original heat map to choose the best approximate match which was nearest neighbor interpolation. Created using the griddata function in Python.

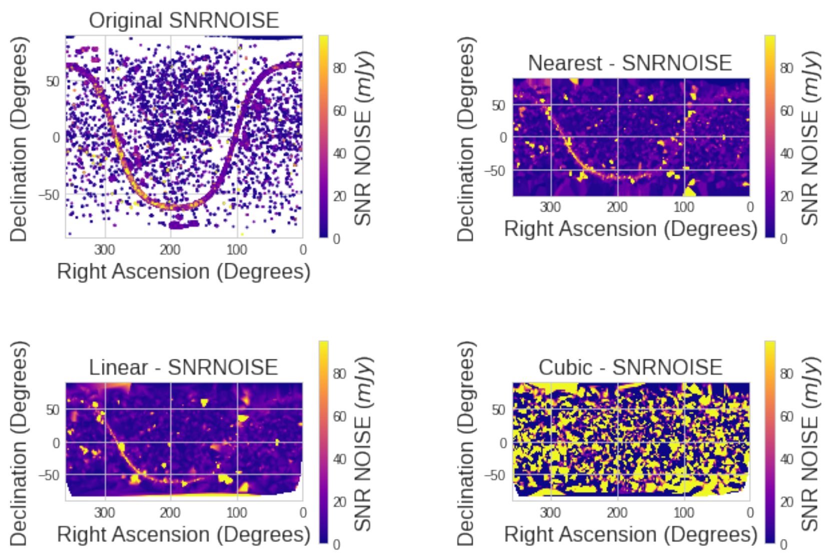

Figure 4: The different methods of interpolation tested for SNR noise. The various methods were compared to the original heat map to choose the best approximate match which was nearest neighbor interpolation. Created using the griddata function in Python.

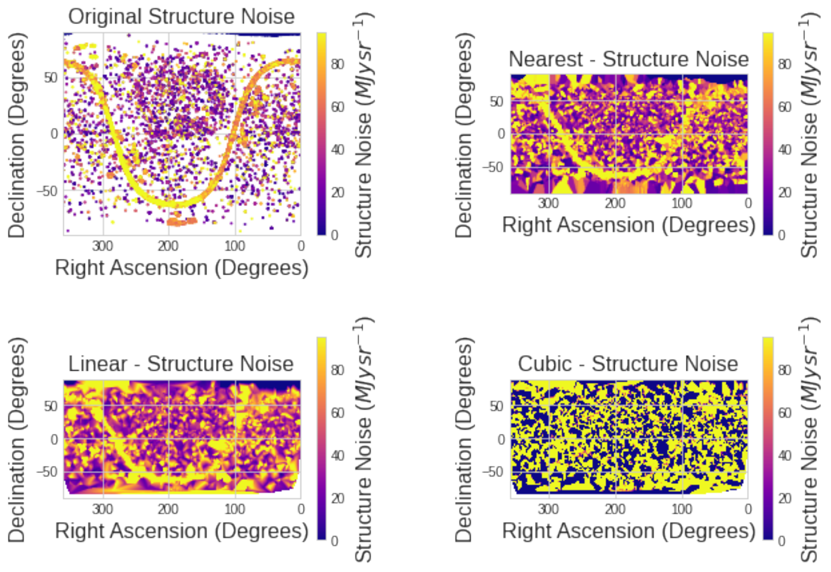

Figure 5: The different methods of interpolation tested for structure noise. The various methods were compared to the original heat map to choose the best approximate match which was nearest neighbor interpolation. Created using the griddata function in Python.

The three-dimensional plots of the separate measures of noise also displayed expected results. Since the range for all of the measures of noise were so high, one would expect there to be many tall spikes and few sorter spikes (Fig. 6). Because the spikes made the graph confusing to view, the natural log of the noise values was taken (Fig. 7).

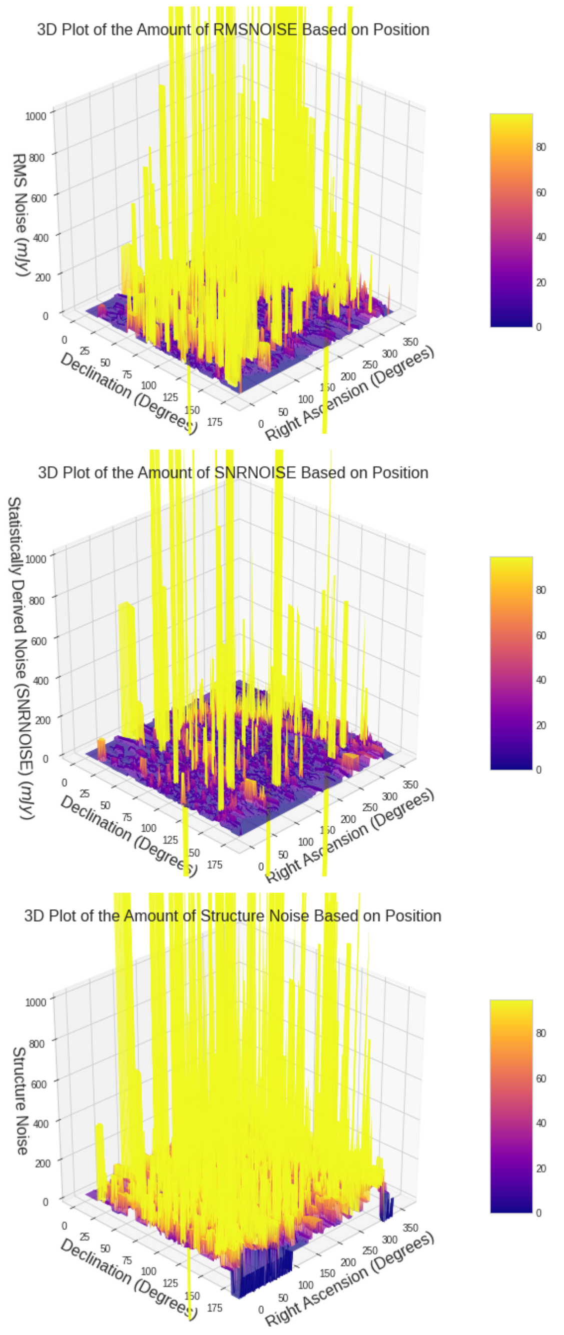

Figure 6: The original three-dimensional triangular surface plots of the interpolated noise in a given position in space.

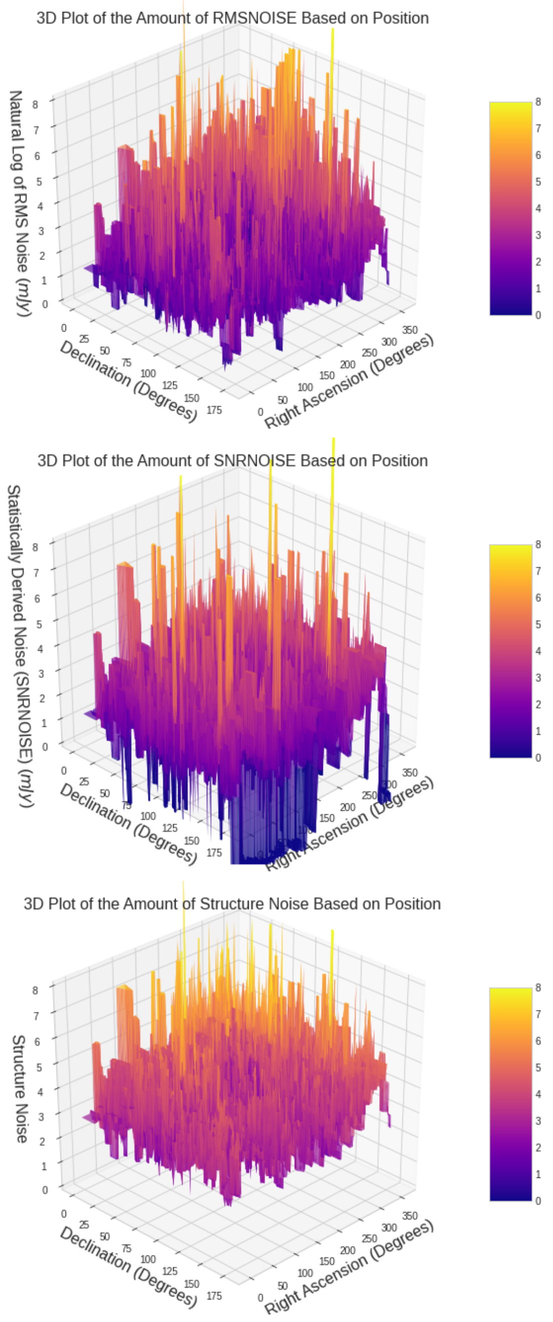

Figure 7: The three-dimensional triangular surface plots of the natural log of the interpolated noise measures in a given position in space. Due to the large outliers in the original version, the natural log was taken to visualize the data better.

Unfortunately, due to time constraints, a function that accepts floats was not created and the objective was not achieved. However, a function *was* created for integers. This was the full extent to which the research was done.

6 Conclusion

This study focused on the interpolation of telescopic noise where no data was captured in order to create a model and function of how much noise one could expect in a given position in the universe. Most noise was centered around dusty areas of the Milky Way galaxy, distant nebulae and bright stars. Future research is required to create a function that can accept float parameters as the function created in this study was limited to integer parameters.

7 Acknowledgements

The Herschel spacecraft was designed, built, tested, and launched under a contract to ESA managed by the Herschel/Planck Project team by an industrial consortium under the overall responsibility of the prime contractor Thales Alenia Space (Cannes), and including Astrium (Friedrichshafen) responsible for the payload module and for system testing at spacecraft level, Thales Alenia Space (Turin) responsible for the service module, and Astrium (Toulouse) responsible for the telescope, with in excess of a hundred subcontractors.

PACS has been developed by a consortium of institutes led by MPE (Germany) and including UVIE (Austria); KU Leuven, CSL, IMEC (Belgium); CEA, LAM (France); MPIA (Germany); INAF-IFSI/OAA/OAP/OAT, LENS, SISSA (Italy); IAC (Spain). This development has been supported by the funding agencies BMVIT (Austria), ESA-PRODEX (Belgium), CEA/CNES (France), DLR (Germany), ASI/INAF (Italy), and CICYT/MCYT (Spain).

References

[1] GL Pilbratt, JR Riedinger, T Passvogel, G Crone, D Doyle, U Gageur, AM Heras, C Jewell, L Metcalfe, S Ott, et al. Herschel space observatory-an esa facility for far-infrared and submillimetre astronomy. *Astronomy & Astrophysics*, 518:L1, 2010.

[2] Micha Schmidt and Frank Keck. The end of life operations of the herschel space telescope. In *SpaceOps 2014 Conference*, page 1935, 2014.

[3] Albrecht Poglitsch, Christoffel Waelkens, Norbert Geis, Helmut Feuchtgruber, Bart Vandenbussche, Louis Rodriguez, Oliver Krause, Etienne Renotte, Christiaan Van Hoof, P Saraceno, et al. The photodetector array camera and spectrometer (pacs) on the herschel space observatory. *Astronomy & astrophysics*, 518:L2, 2010.

[4] G Marton, L Calzoletti, AM Garcia, C Kiss, R Paladini, B Altieri, M Sánchez Portal, M Kidger, et al. The herschel/pacs point source catalogue explanatory supplement. *arXiv preprint arXiv:1705.05693*, 2017.

[5] D.S. Birney. *Observational Astronomy*. Cambridge University Press, 1991.

[6] Paul Daniel Dumitru, Marin Plopeanu, and Dragos Badea. Comparative study regarding the methods of interpolation. *Recent advances in geodesy and Geomatics engineering*, 1:45–52, 2013.

[7] G Aniano, Bruce T Draine, Daniela Calzetti, DA Dale, CW Engelbracht, KD Gordon, LK Hunt, RC Kennicutt, O Krause, AK Leroy, et al. Modeling dust and starlight in galaxies observed by spitzer and herschel: Ngc 628 and ngc 6946. *The Astrophysical Journal*, 756(2):138, 2012.

[8] Paul Rosen, Anil Seth, Betsy Mills, Adam Ginsburg, Julia Kamenetzky, Jeff Kern, Chris R Johnson, and Bei Wang. Using contour trees in the analysis and visualization of radio astronomy data cubes. *arXiv preprint arXiv:1704.04561*, 2017.

[9] Ekaba Bisong. *Building machine learning and deep learning models on Google Cloud platform: a comprehensive guide for beginners*. Apress, 2019.

A Appendices

A.1 Code

```
"""FinalHerschelProject.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/15DhJydhVVIwIS0iS2 \
    cHasaAGzluN0Ojp
"""

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from scipy import interpolate
from scipy.interpolate import griddata
from mpl_toolkits import mplot3d
plt.style.use('seaborn-whitegrid')

from google.colab import drive
drive.mount('/content/drive')

data = pd.read_csv('/content/drive/My Drive/HERSCHEL/HPPSC_070 \
    _v1.csv')

ra = data[["ra"]].values
dec = data[["dec"]].values
rms = data[["rms"]].values
snrnoise = data[["snrnoise"]].values
strn = data[["strn"]].values

"""# Heat Maps

## RMS NOISE HEATMAP
"""

## RMS NOISE HEAT MAP
plt.figure(figsize = (10, 5))

# rms = rms.to_numpy()

plot = plt.scatter(ra, dec, c = rms, cmap = "plasma", s = 1)
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.xlim(0, 360)
plt.ylim(-90, 90)
plt.gca().invert_xaxis()
plt.title("RMS Noise Based on Position in Space")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.gcf().set_size_inches(15, 8)
plt.show()


## SNR NOISE HEAT MAP
plt.figure(figsize = (10, 5))

# snrnoise = snrnoise.to_numpy()

plot = plt.scatter(ra, dec, c = snrnoise, cmap = "plasma", \
     s = 1)
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,95)
plt.xlim(0, 360)
plt.ylim(-90, 90)
plt.gca().invert_xaxis()
plt.title("Statistically Derived Noise Based on Position \
    in Space")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
plt.gcf().set_size_inches(15, 8)
plt.show()

"""## STRN NOISE HEATMAP"""

## STRN NOISE HEAT MAP
plt.figure(figsize = (10, 5))

# strn = strn.to_numpy()

plot = plt.scatter(ra, dec, c = strn, cmap = "plasma", \
     s = 1)
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
plt.clim(0,95)
plt.xlim(0, 360)
plt.ylim(-90, 90)
plt.gca().invert_xaxis()
plt.title("Structure Noise Based on Position in \
         Space")
plt.xlabel("Right Ascension")
plt.ylabel("Declination")
plt.gcf().set_size_inches(15, 8)
plt.show()
```

```
"""# Interpolation"""

ra = ra.flatten()
dec = dec.flatten()
rms = rms.flatten()
snrnoise = snrnoise.flatten()
strn = strn.flatten()

"""## Interpolation Exploration

### RMS
"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]

grid_z0_rms_nearest = griddata((ra, dec), rms, \
                     (xi, yi), method='nearest')
grid_z1_rms_linear = griddata((ra, dec), rms, \
                     (xi, yi), method='linear')
grid_z2_rms_cubic = griddata((ra, dec), rms, \
                    (xi, yi), method='cubic')


plt.subplot(221)
plot = plt.scatter(ra, dec, c = rms, cmap = "plasma", \
     s = 1)
plt.xlim(0, 360)
plt.ylim(-90, 90)
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title("Original RMS")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
# plt.show()

plt.subplot(222)
plt.imshow(grid_z0_rms_nearest.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 10)
plt.title('Nearest - RMS')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
plt.subplot(223)
plt.imshow(grid_z1_rms_linear.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 10)
plt.title('Linear - RMS')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.subplot(224)
plt.imshow(grid_z2_rms_cubic.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
plt.title('Cubic - RMS')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 10)

plt.show()

"""### SNRNOISE"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]

grid_z0_snrnoise_nearest = griddata((ra, dec), \
    snrnoise, (xi, yi), method='nearest')
grid_z1_snrnoise_linear = griddata((ra, dec), \
    snrnoise, (xi, yi), method='linear')
grid_z2_snrnoise_cubic = griddata((ra, dec), \
    snrnoise, (xi, yi), method='cubic')


plt.subplot(221)
plot = plt.scatter(ra, dec, c = snrnoise, cmap = "plasma", \
    s = 1)
plt.xlim(0, 360)
plt.ylim(-90, 90)
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title("Original SNRNOISE")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
# plt.show()

plt.subplot(222)
plt.imshow(grid_z0_snrnoise_nearest.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('Nearest - SNRNOISE')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.subplot(223)
plt.imshow(grid_z1_snrnoise_linear.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('Linear - SNRNOISE')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.subplot(224)
plt.imshow(grid_z2_snrnoise_cubic.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
plt.title('Cubic - SNRNOISE')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,80)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)

plt.show()

"""### STRN NOISE"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]

grid_z0_strn_nearest = griddata((ra, dec), \
    strn, (xi, yi), method='nearest')
grid_z1_strn_linear = griddata((ra, dec), \
    strn, (xi, yi), method='linear')
grid_z2_strn_cubic = griddata((ra, dec), \
    strn, (xi, yi), method='cubic')


plt.subplot(221)
plot = plt.scatter(ra, dec, c = strn, \
    cmap = "plasma", s = 1)
plt.xlim(0, 360)
plt.ylim(-90, 90)
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title("Original Structure Noise")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
# plt.show()

plt.subplot(222)
plt.imshow(grid_z0_strn_nearest.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('Nearest - Structure Noise')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.subplot(223)
plt.imshow(grid_z1_strn_linear.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('Linear - Structure Noise')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.subplot(224)
plt.imshow(grid_z2_strn_cubic.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
plt.title('Cubic - Structure Noise')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
```

```
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)

plt.show()

"""## RMS NOISE INTERPOLATED (HEATMAP)"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]

grid_z0_rms = griddata((ra, dec), rms, (xi, yi), \
    method='nearest')

plt.imshow(grid_z0_rms.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("RMS ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('RMS NOISE Interpolated (Nearest)')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.show()

"""## SNRNOISE INTERPOLATED HEATMAP"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]
grid_z0_snrnoise = griddata((ra, dec), \
    snrnoise, (xi, yi), method='nearest')

plt.imshow(grid_z0_snrnoise.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("SNR NOISE ($mJy$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('SNR NOISE Interpolated (Nearest)')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")


plt.show()

"""## STRN NOISE INTERPOLATED HEATMAP"""

xi,yi = np.mgrid[0:360:360j, -90:90:180j]
grid_z0_strn = griddata((ra, dec), strn, \
    (xi, yi), method='nearest')

plt.imshow(grid_z0_strn.T, origin='lower', \
    cmap = "plasma", extent=[0, 360, -90, 90])
cbar = plt.colorbar(plot)
cbar.set_label("Structure Noise ($MJy sr^{-1}$)")
plt.clim(0,95)
plt.gca().invert_xaxis()
plt.gcf().set_size_inches(15, 8)
plt.title('Structure Noise Interpolated (Nearest) ')
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")

plt.show()

grid_z0_strn[250][80]

"""#3D Maps of Interpolation

## RMS NOISE INTERPOLATED 3D MAP
"""

x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_rms[i][j])

fig = plt.figure()
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, z, cmap="plasma", \
    linewidth=0.01, vmin=0, vmax=95)
ax.set_zlim3d(0, 1000)
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of RMSNOISE \
    Based on Position")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
ax.set_zlabel('RMS Noise ($mJy$)')
plt.show()

"""## SNRNOISE INTERPOLATED 3D MAP"""
```

```
x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_snrnoise[i][j])

fig = plt.figure()
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, z, cmap="plasma", \
    linewidth=0.01, vmin=0, vmax=95)
ax.set_zlim3d(0, 1000)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of SNRNOISE Based on \
    Position")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
ax.set_zlabel('Statistically Derived Noise (SNRNOISE) \
    ($mJy$)')
plt.show()

"""## STRN NOISE INTERPOLATED 3D MAP"""

x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_strn[i][j])

fig = plt.figure()
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, z, cmap="plasma", \
    linewidth=0.01, vmin=0, vmax=95)
ax.set_zlim3d(0, 1000)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of Structure \
    Noise Based on Position")
plt.xlabel("Right Ascension (Degrees)")
plt.ylabel("Declination (Degrees)")
ax.set_zlabel('Structure Noise')
plt.show()

"""# 3D Maps of Interpolation (Natural Log)

## RMS NOISE INTERPOLATED LOG 3D MAP
"""

x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_rms[i][j])

fig = plt.figure()
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, np.log(z), cmap="plasma" \
    , linewidth=0.01, vmin=0, vmax=8)
ax.set_zlim3d(0, 8)
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of RMSNOISE \
    Based on Position", fontsize=16)
plt.xlabel("Right Ascension (Degrees)", fontsize=16)
plt.ylabel("Declination (Degrees)", fontsize=16)
ax.set_zlabel('RMS Noise ($mJy$)', fontsize=16)
plt.show()

"""## SNRNOISE INTERPOLATED LOG 3D MAP"""

x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_snrnoise[i][j])

fig = plt.figure()
```

```
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, np.log(z), cmap="plasma", \
    linewidth=0.01, vmin=0, vmax=8)
ax.set_zlim3d(0, 8)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of SNRNOISE \
    Based on Position", fontsize=16)
plt.xlabel("Right Ascension (Degrees)", fontsize=16)
plt.ylabel("Declination (Degrees)", fontsize=16)
ax.set_zlabel('Statistically Derived Noise (SNRNOISE) \
    ($mJy$)', fontsize=16)
plt.show()

"""## STRN INTERPOLATED LOG 3D MAP"""

x = []
y = []
z = []
for i in range(0, 360):
  for j in range(0, 180):
    x.append(i)
    y.append(j)
    z.append(grid_z0_strn[i][j])

fig = plt.figure()
ax = mplot3d.Axes3D(fig)
surf =  ax.plot_trisurf(x, y, np.log(z), cmap="plasma" \
    , linewidth=0.01, vmin=0, vmax=8)
ax.set_zlim3d(0, 8)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.gca().invert_xaxis()
ax.view_init(30, 45)
plt.gcf().set_size_inches(8, 6)

plt.title("3D Plot of the Amount of Structure Noise \
    Based on Position", fontsize=16)
plt.xlabel("Right Ascension (Degrees)", fontsize=16)
plt.ylabel("Declination (Degrees)", fontsize=16)
ax.set_zlabel('Structure Noise', fontsize=16)
plt.show()
```