Reconstructing 2D Cosmic Dust Maps from Scattered Observations

AM 205 Final Project, Harvard University

Stephen Slater, Zizheng Xu, Kenneth Chen

December 17, 2018

Abstract

The Milky Way contains cosmic dust, which impacts optical and infrared observations of certain regions. In particular, the dust scatters and absorbs optical, near-infrared, and ultraviolet light from behind it, and therefore affects observations of distant objects, such as stars. Creating a 2D map of cosmic dust over the whole sky helps correct reddening effects and thus recover unbiased light spectra of distant extra-galactic objects. By comparing the observed spectrum of a star and its theoretical spectrum, we can get the extinction caused by dust that is in the direction of that star. Extinction is a measurement of the amount of dust present. There are two notable challenges to constructing the dust map accurately. First, we only have data where stars are located, so we have to figure out the dust distribution in areas that are between the stars. Astronomers might want to query the dust extinction in the regions between measurements in order to correct their observed light spectra of distant objects. Second, in real measurements of dust, errors are non-negligible, which makes it more difficult to reconstruct the distribution of dust. To create a dust map in light of these challenges, we present two methods: the inverse Lanczos interpolation and the least squares polynomial fit. To test our algorithms, we compare the reconstructed dust map to a known dust distribution. We find that when the sample points are sparse and the noise is large, the inverse Lanczos interpolation method performs better than the polynomial fitting method, which implies that the Lanczos method is robust to the two challenges of unknown star positions and noisy measurements.

1 Scientific Background

Interstellar dust is one kind of interstellar medium in the Milky Way, usually consisting of silicate particles, graphite, and large hydrocarbon molecules (Draine 2003 [1]). It scatters and absorbs optical, near-infrared, and ultraviolet light behind it. This effect is called extinction. The light spectrum is "reddened" after penetrating dust, due to absorption laws of physics. In addition, dust thermal emissions bias cosmic microwave background observations. In extra galactic and cosmological observations, we often want to correct for the extinction effect. We can properly compute this correction provided that we know how much dust the light passed through. Therefore, creating a map of the dust extinction is a critical problem in astronomy.

In practice, we can detect and measure the existence of cosmic dust either (1) directly from its infrared thermal emissions or (2) via the extinction effect that it imposes on the light spectra of stars behind it. Previous work in SFD [2] focuses on the first method, and has been made into a Python package that supports user queries. To map the dust, the researchers recorded dust emission from infrared images. With due processing, the dust map can be constructed from those images. However, limited by the aperture of telescopes, that map has a point spread function (PSF) of 6

arcminutes. This means that any fine dust structure smaller than 6 arcminutes in the sky cannot be well represented in that map.

With spectral energy distribution measurements from the Pan-STARRS (Flewelling et al. (2016) [3]) and SDSS (Blanton et al. (2017) [4]), we can calculate the dust extinction value particular to the position of a star. Ideally, the PSF can reach approximately tens of arcseconds, depending on the angular distances between stars, which provides a finer, more detailed distribution of dust.

Next, we will explain the methods we developed to construct data in order to test our algorithms.

2 Data Collection from the SFD Dust Map

2.1 Benchmark Image

We identified a region of interest in the sky and selected a 30×30 image that had interesting features and contrast. Then, we queried the SFD map every 2.4 arcminutes (width of 1 pixel) to create a 30×30 pixel image and call this image $\tilde{\mathbf{E}}$, as shown in Figure 1.

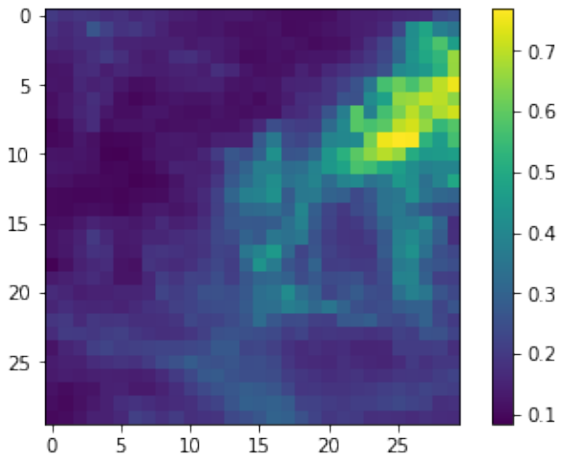

Figure 1: The 30×30 benchmark image.

2.2 Creating Mock Observational Data

We can assume that stars are uniformly distributed in the area that we construct our image. In a simulation of real observations, we get data from random positions in the SFD map. For each observation, we record the coordinates and the dust extinction value.

In the square area of the benchmark image, we sampled $100 * 30^2$ data points at random 2D positions, which gives us 100 sample points on average, per pixel. We queried the SFD data using the Python module `dustmaps`[6]. In the following figure, we demonstrate different amounts of sampled data:

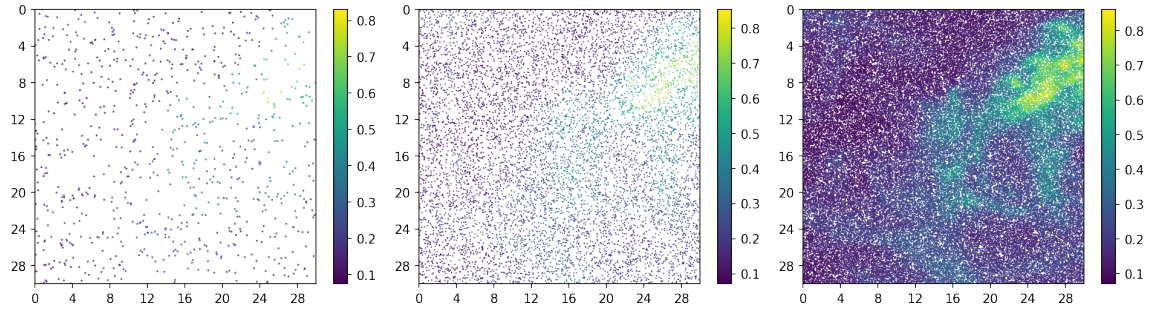

Figure 2: Left: 900 points sampled, 1 per pixel on average; middle: 10 points per pixel on average; right: 100 points per pixel on average.

3 Mathematical Formulation of the Problem

3.1 Notation

Consider a region of size $W \times W$ from which we receive NW^2 uniformly distributed observation points. Thus, N is the average number of points per 1x1 region. Our observations of dust are recorded as $(\mathbf{X}, \mathbf{Y}, \mathbf{E}) \equiv (x_k, y_k, e_k)|_{k=1}^{NW^2}$, where (x_k, y_k) are the coordinates and $e_k \in [0, 1]$ is the value of the observed local dust extinction at position (x_k, y_k). We want to reconstruct an image $\hat{y} \in \mathbb{R}^{W \times W}$ so that it is as close as possible to the original distribution of dust extinction $\tilde{\mathbf{E}}$ within the square area of $[0, W] \times [0, W]$.

To solve this problem, we explore the options of inverse Lanczos 2D interpolation and 2D polynomial fitting.

3.2 Error Evaluation

We evaluate the accuracy of the reconstructed figure based on root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{W-1} \sum_{j=0}^{W-1} (\hat{y}_{ij} - \tilde{\mathbf{E}}_{ij})^2}{W^2}}$$

where $\hat{y}$ is an estimate of the map, $\hat{y}_{ij}$ is the estimate at point (ij); and $\tilde{\mathbf{E}}_{ij}$ is the true value at the grid point (i, j).

We will see later that the Lanczos method will be less accurate near the margins of the images; the width of the inaccurate region at the margins equals the order of the Lanczos function, a. When we calculate RMSE, we disregard the a margin of width $a = 3$ pixels, in order to make a fair comparison between the family of Lanczos methods (with maximum order 3) and the family of polynomial fitting methods. The RMSE is reduced to:

$$\text{RMSE}_{reduced} = \sqrt{\frac{\sum_{i=a}^{W-1-a} \sum_{j=a}^{W-1-a} (\hat{y}_{ij} - \tilde{\mathbf{E}}_{ij})^2}{(W - 2a)^2}}$$

For the rest of the paper, when we refer to RMSE, we are referring to the reduced version.

4 The Inverse Lanczos Interpolation Method

4.1 Lanczos Function

In computer graphics, Lanczos interpolation is a method of interpolating a grid into a smooth surface. The theoretical support for this method is that the Lanczos function is an approximation to the sinc function[5], which can perfectly reconstruct a 2D surface from grid points, as long as the Nyquist sampling criterion is met.

The Lanczos function is defined as follows:

$$L(x) = \begin{cases} 1 & x = 0 \\ \frac{a \sin(\pi x) \sin(\pi x/a)}{\pi^2 x^2} & -a \leq x < a \text{ and } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where a is a chosen positive integer, which determines where we truncate the tail of the sinc function[7]. The plot of this function in one dimension for $a = 2, 3$ is as follows:

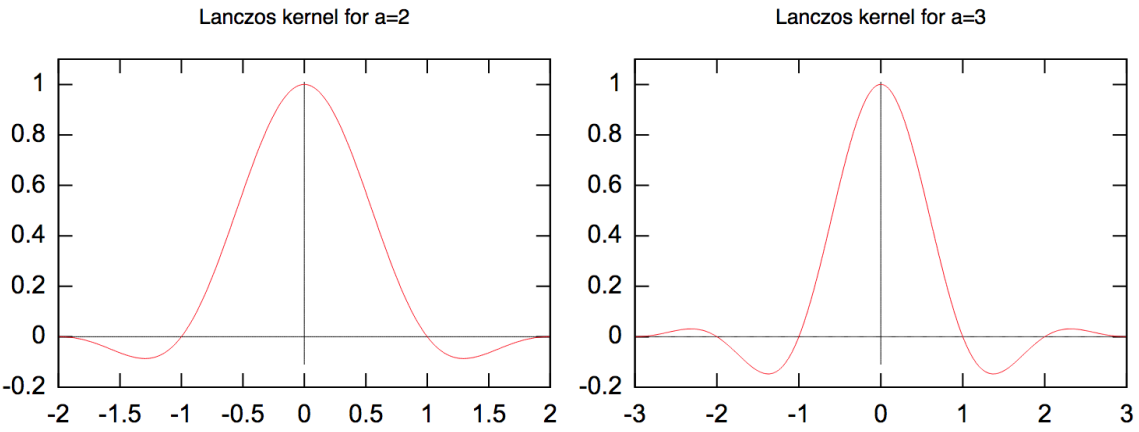

Figure 3: Lanczos kernel for $a = 2, 3$.

4.2 Lanczos Interpolation

With the parameter a fixed, the 1D Lanczos function can be used to reconstruct a curve given equally spaced grid points. The interpolation function is:

$$S(x) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} s_i L(x - i)$$

In a 2D setup, the Lanczos function is defined as:

$$L(x, y) = L(x)L(y)$$

The 2D Lanczos function preserves the property that its value goes to zero when x and y are integers, except when x and y are zeroes in which case $L(0, 0) = 1$. This is because the 1D Lanczos function

is 0 when x is an integer such that $-a \leq x < a, x \neq 0$ due to the sin operation. For this reason, we can still use the Lanczos function to compute 2D interpolation.

Using this function to interpolate a surface from known 2D grid points c_{ij}, we get

$$S(x_k, y_k) = \sum_{i=\lfloor x_k \rfloor - a + 1}^{\lfloor x_k \rfloor + a} \sum_{j=\lfloor y_k \rfloor - a + 1}^{\lfloor y_k \rfloor + a} c_{ij} L(x_k - i) L(y_k - j) \tag{1}$$

where $\lfloor x \rfloor$ means to floor-round a real number to an integer.

Using this formula, we can use the grid points c_{ij} to estimate the value e_k at point (x_k, y_k). For example, when $a = 1$, the value e_k is a combination of these grid values:

$$c_{\lfloor x_k \rfloor, \lfloor y_k \rfloor}, c_{\lfloor x_k \rfloor + 1, \lfloor y_k \rfloor}, c_{\lfloor x_k \rfloor, \lfloor y_k \rfloor + 1}, c_{\lfloor x_k \rfloor + 1, \lfloor y_k \rfloor + 1}$$

where weights are given by the corresponding Lanczos function values:

$$L(x_k - \lfloor x_k \rfloor) L(y_k - \lfloor y_k \rfloor),\ L(x_k - (\lfloor x_k \rfloor + 1)) L(y_k - \lfloor y_k \rfloor),$$
$$L(x_k) L(y_k - (\lfloor y_k \rfloor + 1)),\ L(x_k - (\lfloor x_k \rfloor + 1)) L(y_k - (\lfloor y_k \rfloor + 1)).$$

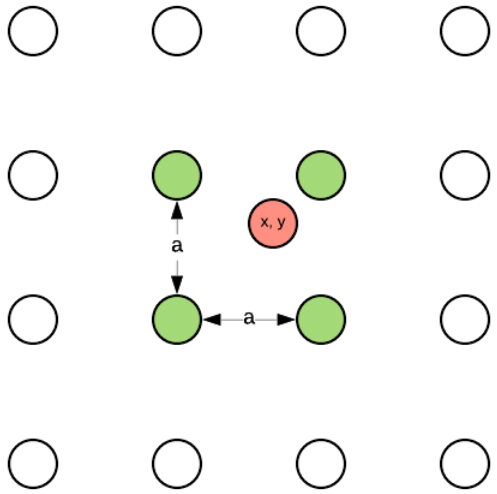

Figure 4: The point (x_k, y_k, e_k) is depicted in red. The image grid points involved in this expression are depicted in green.

We are interested in the inverse of this problem. Rather than use grid points to estimate the randomly-positioned values e_k, we will use values e_k to estimate the grid points. To achieve this, we should construct the Lanczos kernel matrix.

4.3 Lanczos Kernel Construction

Let us assume that we have equally-spaced grid data c_{ij} and we have to give values to NW^2 observation data points that might be anywhere in the map. Let the image width be W. This means

that we have on average N observations per grid point, where we consider the grid points to be the pixels of the $W \times W$ image.

We then have the vector of extinction values as $\mathbf{E} \in \mathbb{R}^{NW^2}$. We solve for all W^2 coefficients c_{ij}, $0 \leq i, j \leq W-1$, in a flattened vector $c \in \mathbb{R}^{W^2}$. Finally, we construct the Lanczos kernel matrix $\mathbf{A}$ as follows:

For the kth data point (x_k, y_k, e_k), where $e_k \in \mathbf{E}$, we apply the interpolation formula in two dimensions from Eq. 1. Let $f(i,j)$ be a bijection from coordinate $(i,j) \in \mathbb{R}^{W \times W}$ to $t \in \mathbb{R}^{W^2}$, i.e. $f(i,j) = iW + j$. Then we have an index t into the flattened coefficients vector c. Now, for the corresponding i, j, we compute $L(x_k - i)L(y_k - j)$ and store this value at $\mathbf{A}[k][f(i,j)]$.

As an example, consider the data point $(x_k, y_k, e_k) = (16.4, 25.0, 0.5)$[1]. We construct the kth row of $\mathbf{A}$, using $a = 2$, as follows:

- Initialize $A[k] = [0, \ldots, 0] \in \mathbb{R}^{W^2}$
- $\lfloor x_k \rfloor = 16$, $\lfloor y_k \rfloor = 25$
- for $i \in \{15, \ldots, 18\}$ and for $j \in \{24, \ldots, 27\}$:

$$\mathbf{A}[k][f(i,j)] = L(x_k - i)L(y_j - j)$$

As a result, we have $A[k] \in \mathbb{R}^{W^2}$ as a row vector with $n_2 = 16$ entries $L(x_k - i)L(y_k - j)$ for the corresponding values of i, j.

In our experiments, we use $a = 1, 2, 3$. Let $n_a = (2a)^2$ refer to the number of grid points used in the boundary box surrounding a point (x_k, y_k). For $a = 1$, this means that each sample is predicted as a linear combination of $n_1 = 4$ Lanczos kernel evaluations, which are weighted by the "true" extinction values c_{ij}, which we solve for using the normal equation to be introduced in Section 4.4. For $a = 2$, we use $n_2 = 16$ points. For $a = 3$, we use $n_3 = 36$ points. For samples near the boundaries of the image, we consider only the $n \leq n_a$ points that are valid pixel indices, which is why use the reduced RMSE as explained in Section 3.2 Specifically, to account for inaccuracies at the boundaries, we compute our results using the inner $(W - 2a)^2$ pixel values, since these are inward enough in the image to be interpolated using all n_a pixels.

In this algorithm, the gridpoint c_{ij} is stored at $c[f(i,j)]$ in the column vector c. Then the matrix-vector product $\mathbf{A}c = \mathbf{E}$ yields the data points e_k, collectively forming the vector $\mathbf{E}$. Now, let's go back to the question setting. We have a set of measurements $(x_k, y_k, e_k)|_{k=1}^{NW^2}$. The equation $\mathbf{A}c = \mathbf{E}$ still holds, and $\mathbf{A}$ can be constructed from $(x_k, y_k)|_{k=1}^{NW^2}$.

4.4 Least Squares Solution to $\mathbf{A}c = \mathbf{E}$

In this section we solve the inverse of the Lanczos interpolation problem, i.e. using interpolated values to solve for grid points. We seek to solve for c in the Lanczos interpolation context. This means we will need to solve an over-determined system of equations if $NW^2 > W^2$, where W^2 is the length of c.

The normal equation gives us the way to find the least squares fit solution to c:

$$\begin{aligned} \mathbf{A}c &= \mathbf{E} \\ c &= (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{E} \end{aligned}$$

This is for the case when our observations are accurate (without noise). Our prediction of the true extinction map $\hat{y}$ is the $W \times W$ reshaped image from the flattened vector c, i.e. $\hat{y}_{ij} = c[f(i,j)]$. Then, we can measure the accuracy of $\hat{y}$ using our RMSE method and the ground truth $\tilde{\mathbf{E}} \in \mathbb{R}^{W \times W}$.

[1] Note that the product of these numbers is 205.

4.5 Examples of Reconstructed Images with Inverse Lanczos Interpolation

We now provide examples of reconstructed images using the inverse Lanczos interpolation method:

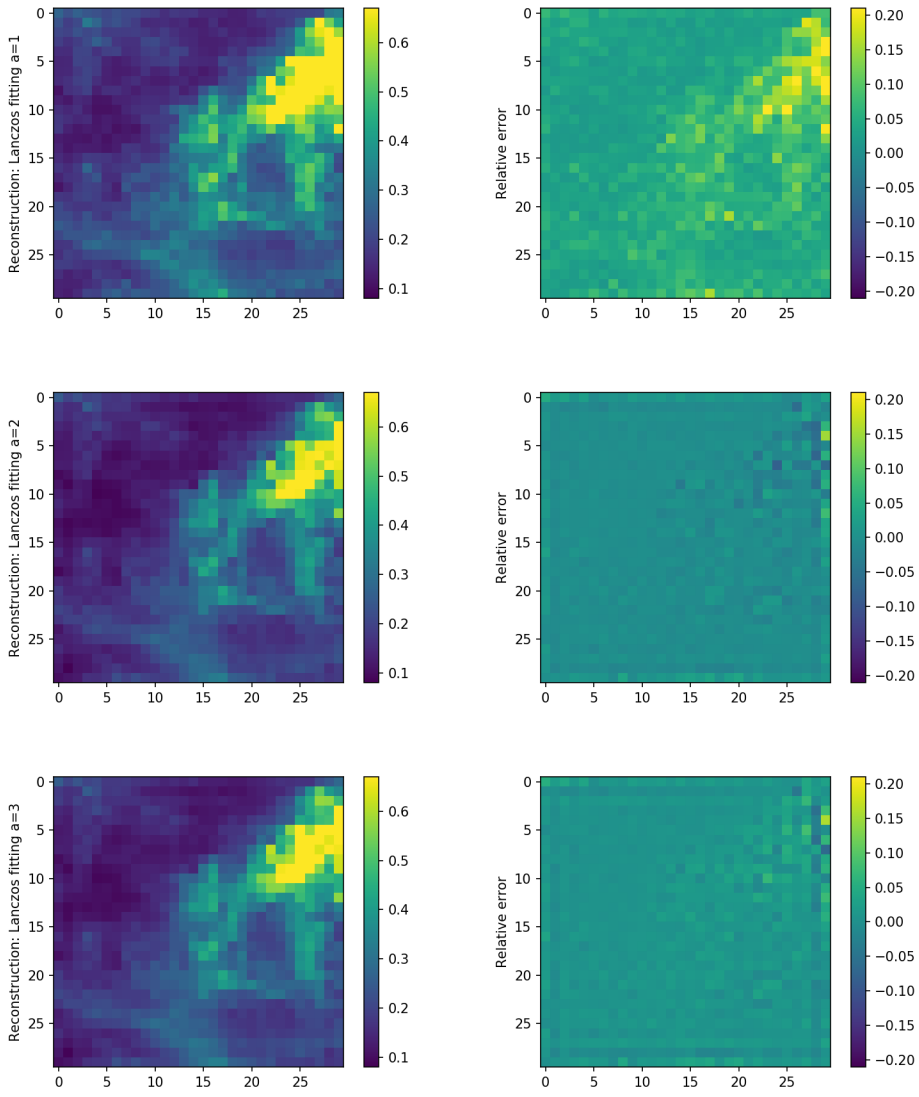

Figure 5: Reconstructions and residue maps compared to the 30×30 benchmark image (Figure 1) for varying values of a in the Lanczos method. In these images, $NW^2 = 11 * (30^2) = 9900$ observations. The two images on the top row show that the Lanczos method for $a = 1$ provides too much weight to the nearest points (within distance 1), and therefore overestimates the level of dust extinction, which is evident in the extra large patch of yellow. The next two rows show that the Lanczos methods with $a = 2, 3$ are more robust, but rely on more points to make each estimate, which explains the boundary of the darker inner square (a pixels in from the border), where points did not receive the complete boundary box as depicted in Figure 4.

5 Polynomial Fitting Method

5.1 Least Squares Fitting

We now formulate the image reconstruction as a least squares problem with local optimization, meaning that instead of fitting a 2D polynomial to all of the observations and reconstructing an image by evaluating the polynomial at each pixel center, we instead fit a 2D polynomial to the roughly N sample points (x', y', e') surrounding each of the $W^2 = 900$ pixel centers (x_c, y_c). The total number of sample points per pixel is approximately N since they are sampled uniformly from the 30×30 image. The samples are shown in Figure 3. In particular, we let each (x_c, y_c) refer to the center point of its own pixel in the 30×30 image. Then, with all of the data points (x', y', e') that fall within the 1x1 grid, we subtract (x_c, y_c) from (x', y') and then scale x' and y' to $[-1, 1]$ (multiply by 2). In other words, from the $100W^2$ samples shown in the rightmost panel in Figure 2, we sample NW^2 and assign each point to its closest pixel center on a 30×30 grid of 1x1 pixels, and then scale the coordinates to $[-1, 1]$.

Now, we have $W^2 = 900$ unique least squares problems, each of which is formulated as follows:

$$\begin{aligned} \mathbf{A}c &= \mathbf{E} \\ c &= (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{E} \end{aligned}$$

Since this is a least squares problem, the setup looks identical to the inverse Lanczos interpolation method described in 4.4, but the meanings are slightly different. In the polynomial fitting case, $\mathbf{A}$ is a Vandermonde matrix, c are the coefficients of a polynomial $p(x, y)$ that we want to solve for in our fitting method, and $\mathbf{E}$ are the dust extinction values for the approximately N sampled points within each pixel. Whereas the Lanczos method solved for each value of the image in the flattened vector c using the Lanczos kernel $\mathbf{A}$, the polynomial fitting method solves for the coefficients c_{ij} that are used to create an image by evaluating the following fitted 2d polynomial p:

$$p(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{m} c_{ij} x^i y^i \tag{2}$$

where m is the maximum degree of each x, y.

With N total points per pixel and degree m, our 2D Vandermonde matrix (0-indexed) is:

$$\mathbf{A} = \begin{bmatrix} 1 & x_0^0 y_0^1 & \dots & x_0^0 y_0^m & \dots & x_0^m y_0^0 & x_0^m y_0^1 & \dots & x_0^m y_0^m \\ 1 & x_1^0 y_1^1 & \dots & x_1^0 y_1^m & \dots & x_1^m y_1^0 & x_1^m y_1^1 & \dots & x_1^m y_1^m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{N-1}^0 y_{N-1}^1 & \dots & x_{N-1}^0 y_{N-1}^m & \dots & x_{N-1}^m y_{N-1}^0 & x_{N-1}^m y_{N-1}^1 & \dots & x_{N-1}^m y_{N-1}^m \end{bmatrix}$$

Via the equation for c above, this solves for coefficients in the form:

$$c = [c_{00}, c_{01}, \dots, c_{0m}, ..., c_{m0}, c_{m1}, \dots, c_{mm}]$$

where $c \in \mathbb{R}^{(m+1)^2}$ and $\hat{y} = c_{00}$, where $\hat{y}$ is our estimate for the value of this pixel (taken to be at the center), since each pixel center is at $(0, 0)$ of the best-fit polynomial. This avoids needing to use the remaining coefficients in c, since the remaining terms in $p(x, y)$ evaluate to 0 when x or y is 0. However, the remaining coefficients could be used if one were to explore the use of other evaluations of p within the grid, i.e. making an estimate based on the average of 4 points surrounding $(0, 0)$.

5.2 Examples of Reconstructed Images using 2D Polynomial Fitting

We now provide examples of image reconstruction using 2D polynomial fitting. The reconstructed images are on the left, and the relative error compared to the baseline image (Figure 1) is shown on the right panel of Figures 6 and 7.

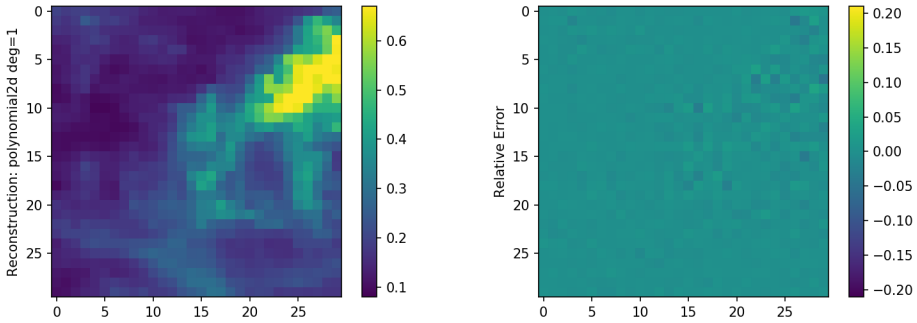

Figure 6: Reconstruction using $N = 11$ points per pixel and 2D polynomial fitting of degree 1. The reconstruction is very accurate.

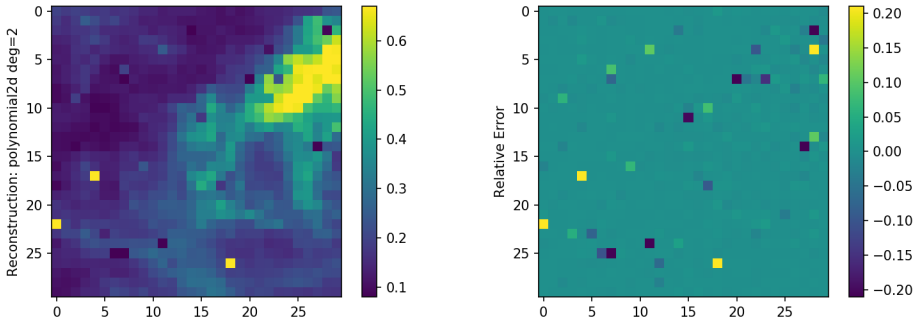

Figure 7: Reconstruction using $N = 11$ points per pixel and 2D polynomial fitting of degree 2. In this case, there are sporadic pixels with very high error. This can occur when N approaches the theoretical lower bound for how many samples per pixel must be required in order to fit a 2D polynomial of degree m. We explain this in detail in the following section.

5.3 Limited Support of N using 2D Polynomial Fitting Method

As seen in the previous section, the image reconstruction from polynomial fitting is accurate when the number of points N used to estimate each pixel is large enough compared to the degree of the polynomial m. In the first case, $m = 1$ and $N = 11$. This yields a well-constrained matrix, and we can solve for the coefficients. However, as seen in the second reconstructed image, the technique of locally fitting each pixel based on the samples corresponding to that pixel breaks down in the polynomial method when N is relatively low. In particular, consider a 2D polynomial fit of degree m for both x and y.

$$p(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{m} c_{ij} x^i y^i \tag{3}$$

There are $(m + 1)^2$ terms in this polynomial. Therefore, when constructing the Vandermonde matrix $\mathbf{A} \in \mathbb{R}^{N \times (m+1)^2}$, we require $N \geq (m + 1)^2$ sample points to estimate the value for the corresponding pixel in the image so that the matrix we invert when solving for the least-squares fit is non-singular.

Then for each 2D polynomial of degree m, we have the following theoretical bound and experimental bound on the number of points needed to fit each pixel. Note that since our data points are sampled uniformly from a larger dataset containing 100 samples per pixel (as described in Section 2.1) and then bucketed into regions corresponding to each pixel in the smaller image, it is possible that when sampling NW^2 points that we have at least one pixel of the W^2 pixels without at least N samples, resulting in a singular matrix. Therefore, the experimental lower bound is greater than or equal to the theoretical lower bound. This is a disadvantage when compared to the inverse Lanczos interpolation method, since the Lanczos kernel has a larger support than the polynomial fitting method.

Lower bound	$m = 1$	$m = 2$	$m = 3$	$m = 4$
Theoretical points required N	4	9	16	25
Experimental points required N	6	11	19	32

Table 1: Theoretical and experimental lower bounds on the number of points N required to fit each pixel, where NW^2 points are sampled uniformly in desire to place N points in each of the W^2 pixels. On average, each of the W^2 pixels has N points, but due to fluctuations in the number of observations assigned to each pixel caused by random sampling, it is possible to require more points experimentally.

We now provide an explanation for the high error seen in the image reconstruction using a degree 2 polynomial in Figure 7. When we sample NW^2 points from the entire dataset of $100 * 30^2$ observations (explained in Section 2.2), on average we expect each pixel to have N assigned observations. It is possible that one pixel has $N + 1$ and another has $N - 1$, however, in which case we need to sample more than NW^2 points, due to the fact that the sampling is random. However, consider the case when a pixel has N samples, where $N < (m+1)^2$, and we desire to fit a polynomial of degree m. Since $N < (m+1)^2$, we know that $\mathbf{A}^\top \mathbf{A}$ must be rank-deficient, and so we should not able to invert it when we compute c. However, due to the limits of machine precision, it is possible to have an invalid inverse (with extremely high absolute values), as shown here:

```
import numpy as np
A=np.random.random((4,5))
print(A)
ATA=A.T.dot(A)
```

```
[[0.11751022 0.6194855  0.67624191 0.2350693  0.08922641]
 [0.10814767 0.37162096 0.26414893 0.79585322 0.36636705]
 [0.94776371 0.48285619 0.55977684 0.27390866 0.28793712]
 [0.57738731 0.67273654 0.26056531 0.10158496 0.20997944]]
```

```
np.linalg.inv(ATA)
```

```
array([[ 4.55708585e+14,  9.21194040e+14,  6.43006179e+14,
        -5.35711628e+14, -1.12371868e+15],
       [ 9.21194040e+14,  1.86215158e+15,  1.29980755e+15,
        -1.08291653e+15, -2.27154586e+15],
       [ 6.43006179e+14,  1.29980755e+15,  9.07283645e+14,
        -7.55890713e+14, -1.58557042e+15],
       [-5.35711628e+14, -1.08291653e+15, -7.55890713e+14,
         6.29759804e+14,  1.32099588e+15],
       [-1.12371868e+15, -2.27154586e+15, -1.58557042e+15,
         1.32099588e+15,  2.77094553e+15]])
```

Figure 8: Due to machine precision error, it is possible to invert a rank-deficient matrix.

As a result, when N is not sufficiently greater than the theoretical bound, it is possible that some pixels will predict an extreme value for the extinction measurement of a given pixel, which will drastically increase the RMSE.

6 Observation Error

We want to take into account errors in measuring extinction observations because every measurement comes with error; for example, measurement error can be introduced by the instrument or photon noise. Here, we assume that our measurements are conducted at a constant level of relative errors, i.e.:

$$\frac{e_{k,\text{observed}} - e_{k,\text{actual}}}{e_{k,\text{actual}}} \sim \mathcal{N}(0, \sigma^2)$$

Hence,

$$e_{k,\text{observed}} \sim \mathcal{N}(e_{k,\text{actual}}, \sigma^2 e_{k,\text{actual}}^2)$$

In order to simulate this error, we multiply every element e_k in $\mathbf{E}$ by $\epsilon_k \sim N(1, \sigma^2)$ independently, where σ is the noise level. In the mock data (Section 2.2), we define our observations $e_k \in \mathbf{E}$ as true observations, so these are $e_{k,actual}$. When we introduce noise, we introduce the constant relative error between what we observe $e_{k,observed}$ (with the noise) and the original samples $e_{k,actual}$.

To write all the observations collectively in matrix-vector form, we have:

$$\begin{aligned}
\mathbf{A}c = \mathbf{E} &\sim \mathcal{MVN}(\mathbf{E}, \sigma^2 diag(\mathbf{E}^2)) \\
c &\sim (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top (\mathcal{MVN}(\mathbf{E}, \sigma^2 diag(\mathbf{E}^2))) \\
c &\sim \mathcal{MVN}((\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{E}, \sigma^2 (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top diag(\mathbf{E}^2)[(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top]^\top)
\end{aligned}$$

where $\mathbf{E}^2$ is the element-wise square of the vector $\mathbf{E}$.

7 Results

In this section, we present the $\log_{10}$ RMSE of each method compared to $\log_{10} N$, where N is the average number of samples per pixel ranging from 3 to 50. Recall that we sample NW^2 observations from the SFD data set, as explained in Section 2 We also present the complete workflow of recreating a 2D surface: we sample observations, then create the dust map via estimated grid point values, and then Lanczos interpolate the grid points to provide a continuous map. This results in a 2D surface that can be queried at any given coordinate within $[0, W] \times [0, W]$.

7.1 Without Noise

First, we apply our methods to the data set without noise as described in Section 2.2. We present results in the table below the figure for selected values of N.

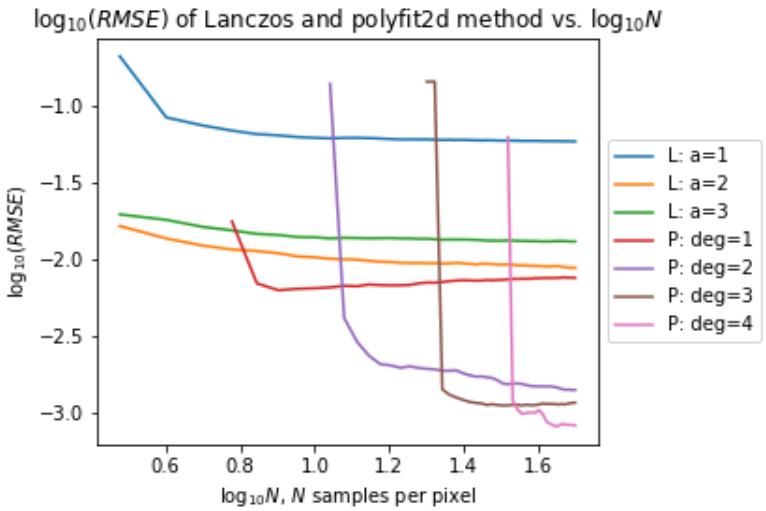

Figure 9: $\log_{10}$ RMSE of image reconstruction using inverse Lanczos interpolation and polynomial fitting for N sample points per pixel for $N = 3, 4, \ldots, 50$.

Method	N=3	N=6	N=10	N=20	N=30	N=40	N=50
Lanczos a=1	0.21150	0.06910	0.06201	0.06055	0.05970	0.05916	0.05873
Lanczos a=2	0.01647	0.01157	0.01030	0.00941	0.00930	0.00902	0.00879
Lanczos a=3	0.01962	0.01543	0.01390	0.01359	0.01323	0.01310	0.01306
Polynomial Degree 1	–	0.01765	0.00647	0.00704	0.00729	0.00751	0.00756
Polynomial Degree 2	–	–	–	0.00193	0.00168	0.00148	0.00140
Polynomial Degree 3	–	–	–	0.14431	0.00113	0.00113	0.00116
Polynomial Degree 4	–	–	–	–	–	0.00104	0.00082

Table 2: RMSE of reconstructed map using observations $\mathbf{E}$ without noise. "–" denotes singular matrix.

7.2 With Small Noise: $\sigma = 0.05$

We introduce small relative error $N(0, \sigma^2)$, where $\sigma = 0.05$, i.e. we independently multiply each observation $e_k \in \mathbf{E}$ by $\mathcal{N}(1, \sigma^2)$.

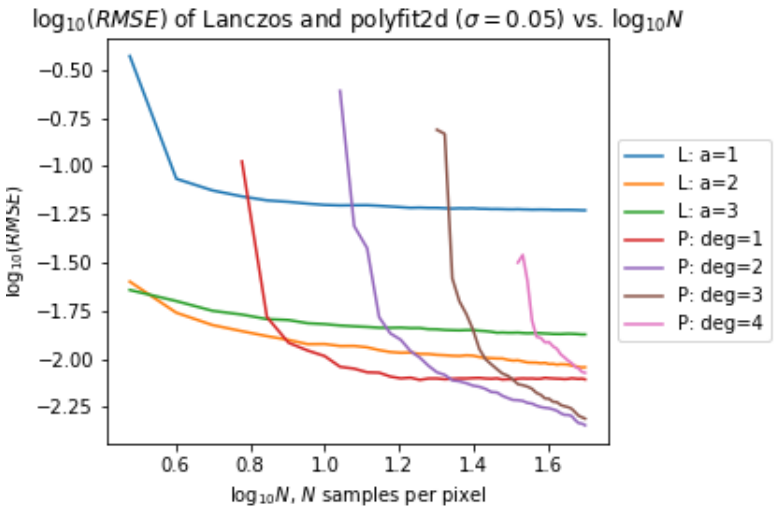

Figure 10: $\log_{10}$ RMSE of image reconstruction with relative error $\mathcal{N}(0, 0.05^2)$ using inverse Lanczos interpolation and polynomial fitting for N sample points per pixel for $N = 3, 4, \ldots, 50$.

Method	N=3	N=6	N=10	N=20	N=30	N=40	N=50
Lanczos a=1	0.37221	0.06961	0.06288	0.06069	0.05981	0.05947	0.05897
Lanczos a=2	0.02516	0.01392	0.01194	0.01050	0.01004	0.00940	0.00905
Lanczos a=3	0.02277	0.01698	0.01515	0.01424	0.01370	0.01352	0.01339
Polynomial Degree 1	–	0.10568	0.01034	0.00791	0.00782	0.00794	0.00783
Polynomial Degree 2	–	–	–	0.00852	0.00643	0.00554	0.00453
Polynomial Degree 3	–	–	–	0.15431	0.00839	0.00621	0.00490
Polynomial Degree 4	–	–	–	–	–	0.01222	0.00847

Table 3: RMSE of reconstructed map using observations $\mathbf{E}$ with relative error $\mathcal{N}(0, 0.05^2)$. "–" denotes singular matrix.

7.3 With Large Noise: $\sigma = 0.30$

We introduce small relative error $N(0, \sigma^2)$, where $\sigma = 0.30$, i.e. we independently multiply each observation $e_k \in \mathbf{E}$ by $\mathcal{N}(1, \sigma^2)$.

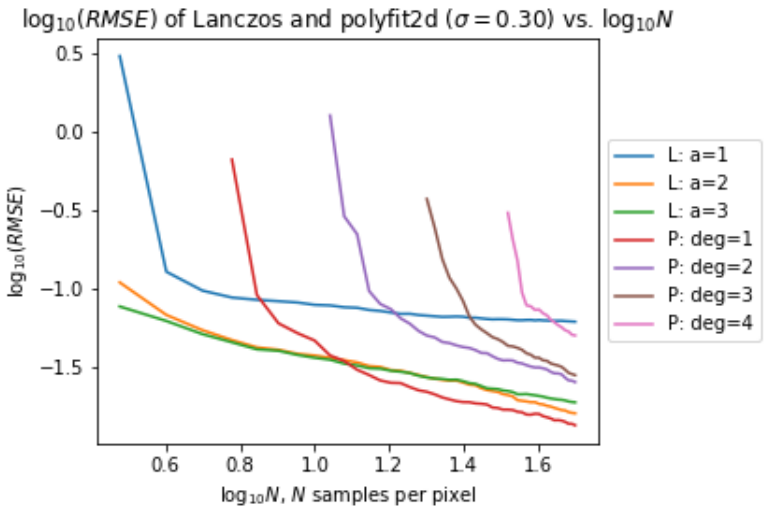

Figure 11: log RMSE of image reconstruction with relative error $\mathcal{N}(0, 0.30^2)$ using inverse Lanczos interpolation and polynomial fitting for N sample points per pixel for $N = 3, 4, \ldots, 50$.

Method	N=3	N=6	N=10	N=20	N=30	N=40	N=50
Lanczos a=1	3.06884	0.08835	0.07934	0.06768	0.06413	0.06348	0.06198
Lanczos a=2	0.11029	0.04729	0.03770	0.02744	0.02235	0.01851	0.01613
Lanczos a=3	0.07745	0.04577	0.03645	0.02732	0.02313	0.02081	0.01895
Polynomial Degree 1	–	0.67060	0.04696	0.02212	0.01756	0.01597	0.01353
Polynomial Degree 2	–	–	–	0.05054	0.03727	0.03163	0.02552
Polynomial Degree 3	–	–	–	0.37548	0.04936	0.03645	0.02823
Polynomial Degree 4	–	–	–	–	–	0.07410	0.05060

Table 4: RMSE of reconstructed map using observations $\mathbf{E}$ with relative error $\mathcal{N}(0, 0.30^2)$. "–" denotes singular matrix.

7.4 Complete Workflow of Map Creation

We now show the complete cycle of recreating a 2D surface using scattered observations. We start with the observations with $N = 10$ points sampled per pixel, as shown in the middle panel of Figure 2.

The middle image is our 2D reconstruction from the samples using the inverse Lanczos interpolation method. We only estimate values at the grid points (pixels), and therefore this image looks pixelated.

The right image is Lanczos interpolated from the middle one. This is the image a user would query from in order to get extinction values at arbitrary coordinates within $[0, W] \times [0, W]$. Notice that the rightmost image has the same grid values as the middle image, but it has continuous values between grid points.

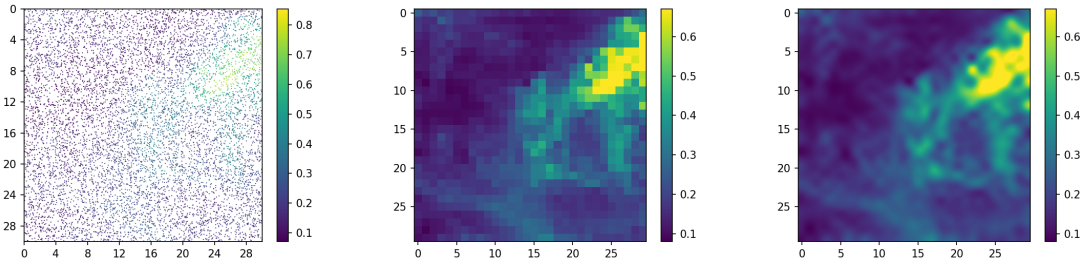

Figure 12: From the sparse observations on the left, we reconstruct the dust map in the middle, which can be interpolated into a continuous map from which astronomers can query in order to receive accurate extinction measurements in unknown regions.

8 Discussion

In this section, we discuss the results of the inverse Lanczos interpolation and polynomial fitting methods, and provide a brief description of other methods that we implemented. We will divide our discussion into subsections to provide maximal clarity to the reader.

8.1 Without Noise

First, we compare the results of the different degrees of each method with the other trials of the same method. In the case of Lanczos, the $a = 2$ method performed better than the $a = 3$ method, and both methods outperformed the $a = 1$ method. This is because the higher order Lanczos methods consider more adjacent points when creating the estimate for each pixel, as explained in Figure 4. Of the polynomial fitting methods, the accuracy is ranked in descending order of degree. However, the higher-order methods are incapable of constructing a map in the case of sparse observations, due to the requirement that each pixel have $N \geq (m+1)^2$ points in order to solve for the coefficients of the 2D polynomial of degree m. For lower N, the polynomial fit of degree 1 is the most useful polynomial fit, and the error is relatively stable due to less expressivity in the model. However, the degree 1 model also reaches a lower bound on N, below which only the Lanczos method is applicable.

Within the range of N where both methods are available, we see that the the polynomial fitting method is more accurate than the Lanczos method. However, when N is small, as explained in Section 5.3, the polynomial method is inaccurate or incapable of solving a system of equations to find the least squares coefficients due to too few observations. Therefore, in the case of no noise and sparse observations (low N), we see that the inverse Lanczos interpolation method is advantageous and that the error is much more stable than in the polynomial case.

8.2 Small Relative Noise: $\sigma = 0.05$

Overall, when we add noise, all of the models decrease in accuracy. In the case of small relative noise, the accuracy of the Lanczos methods maintains the same order: first $a = 2$, then $a = 3$, and then $a = 1$, except at the absolute lowest values of N, in which case $a = 3$ performs the best. However, the polynomial fitting methods respond differently to the noise than without noise. In particular, the degree 4 polynomial is now the least accurate. The degree 3 polynomial is now worse than the degree 2 method, which is the best polynomial method until it reaches close to the lower bound of N, in which case the degree 1 polynomial becomes the most accurate.

As far as comparing the Lanczos results to the polynomial fitting results, we see the same pattern as in the case without noise. The Lanczos $a = 2, 3$ methods are similar in error to the polynomial of degree 1, and maintain a stable level of error as N decreases beyond the limit where local polynomial fitting of each pixel is applicable.

8.3 Large Relative Noise: $\sigma = 0.30$

At high levels of noise, we notice that the higher-degree polynomial fitting methods severely overfitted the noisy observations. Specifically, at every level of N, the accuracy decreases as the degree of the fitted polynomial increases. An explanation for this is that the higher-order polynomials are more complex, and with more expressivity they can fit to the noise, whereas a simpler polynomial model has more bias but is less likely to overfit on noise. Furthermore, the Lanczos $a = 2, 3$ methods are much more on par with the best polynomial method of degree 1 for all values of N. At the largest values of N, the $a = 2$ method outperforms the $a = 3$ method, and vice versa at the smallest values of N. One reason for this could be that when the noise is large and N is very small, it is beneficial to use an $a = 3$ Lanczos kernel, since this provides more examples and could give a more accurate area of pixel weightings on each other, whereas $a = 2$ could not perform as well if there are very few pixels and all of the values are noisy.

8.4 Other Methods

Throughout this project, we considered various methods other than the Lanczos and polynomial methods discussed in this paper in order to reconstruct the dust maps; specifically, we implemented least squares Chebyshev polynomial fitting, polynomial interpolation, and radial basis function approximation. However, since these methods did not perform as well as the Lanczos and polynomial methods in terms of RMSE, we will not describe the theory of these methods in detail in this paper. Nonetheless, we would like to provide an overview of reasons why they did not work well.

We originally thought that using a least squares 2D Chebyshev polynomial fit in each pixel could provide a more accurate fit than the least squares polynomial fit, since the Chebyshev polynomials oscillate in the interval [-1,1]. However, it turns out that the Chebyshev and regular polynomial fittings of the same order yield exactly the same results, since the coefficients of the fitted Chebyshev polynomial can be created from a linear transformation of the coefficients of a regular polynomial. For a polynomial of the same degree, there is one "best fit"—which is given by the closed form for the least squares approximation.

We also implemented direct polynomial interpolation, but the results were poor, since we cannot control the positions of stars (observed points). Randomly positioned interpolation points render interpolation extremely unstable. Furthermore, interpolation methods tend to perform poorly in the case of noisy data when compared to least squares fitting.

We also implemented radial basis function approximation of the 2D surfaces in each pixel using cubic ($\phi(r) = r^3$) and thin plate ($\phi(r) = r^2 \ln r$) kernels to fit the surface, where r is the distance to the center of the assigned pixel, but these proved to not be ideal. In the context of this method, we have no reason to assume every point in the dust distribution affects adjacent areas in a radial-based way.

9 Future Work

9.1 Quality of Smooth 2D Surface

Recall that we measured the quality of the reconstructed 2D surface using RMSE of grid point data and the benchmark image. However, this is an approximation to the L2 norm of the difference of the two distributions: the reconstructed dust map and the "true" dust map.

We could determine the error via the following:

$$\sqrt{\frac{1}{W^2}\int_0^W \int_0^W (I_c - G_c)^2 dxdy}$$

where I_c is our interpolated surface and G_c is the true surface of extinction values at each point in $[0, W] \times [0, W]$.

This integral can be calculated, for example, using a numerical approximation with very small interval changes dx, dy, or alternatively, with 2D quadrature rules.

9.2 Stitching Maps Together

Our algorithms reconstruct regional 2D dust maps from scattered observations, but we still have more work to do make a map of the entire sky. Further work will include stitching smaller maps together into a larger map, while preserving the continuity at the borders between the pieces. In the inverse Lanczos interpolation method, this can be solved by recreating dust maps of adjacent regions that overlap with margins of size $2a$. This way, when we take the inner map of size $(W-2a)^2$ from each adjacent image, we have a fine map of the combined region.

Something to keep in mind is that that the celestial coordinate grid points near the polar area will be squeezed together in longtitude. For example, consider a horizontal line of latitude at 80 degrees and another at 85 degrees. At a higher latitude, a line spanning 10 degrees of longtitude is shorter. Therefore, instead of square images, we should take into account the trapezoidal geometry on the celestial sphere (looking at the sky from the Earth).

10 Conclusion

In this paper, we proposed the inverse Lanczos interpolation method and tested it against 2D polynomial fitting in various conditions of data sparsity and noise. With dense observations and no noise, least squares polynomial fitting is a very accurate method of reconstructing dust maps from scattered observations, and performs better than the Lanczos method. However, when observations are sparse and noise is large, polynomial fitting tends to fit more to noise rather than the true distribution of dust, and will break down if it does not receive enough observation data within one pixel. This is where the inverse Lanczos interpolation method has its advantage. Compared to the polynomial method, it is robust against noise and sparse distributions of observation points, which are the two main challenges astronomers face in real settings.

11 Acknowledgements

We would like to acknowledge Dr. Douglas Finkbeiner, Professor of Astronomy and Physics at Harvard University, for his advice and guidance during this project.

References

[1] Draine, B. T. 2003, ARA&A, 41, 241.

[2] Schlegel, D. J., Finkbeiner, D. P., & Davis, M. 1998, , 500, 525

[3] Flewelling, H. A., Magnier, E. A., Chambers, K. C., et al. 2016, arXiv e-prints , arXiv:1612.05243.

[4] Blanton, M. R., Bershady, M. A., Abolfathi, B., et al. 2017, , 154, 28

[5] Whittaker-Shannon interpolation formula, from Wikipedia.

[6] Python package: `dustmaps`. This is used to access the SFD dataset of cosmic dust. See documentation here.

[7] Lanczos resampling, from Wikipedia: .