VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Antarctic Ice Sheet Visualization

Josh O'Hagan

Supervisors: Taehyun Rhee, Jacob Young

10/10/2021

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours.

Abstract

Rendering a point cloud (a set of points in 3D space) is useful for visualizing real-world data. A higher resolution point cloud gives more accurate rendering. State-of-the-art surface approximation techniques in computer graphics work well to approximate surfaces of point clouds. Points lying on the reconstructed surface can then be added to the point cloud to increase the point cloud resolution. There are many of these techniques, each with its own pros and cons. Increasing the resolution of a point cloud is useful, as sometimes a point cloud is supplied at a resolution that is too low to be rendered accurately.

In this report, I review existing literature regarding surface reconstruction techniques. After choosing a direction for the research, the development of an artifact is then discussed. In particular, a Radial Basis Function (RBF) is used as an interpolation technique to increase the resolution of the data. An evaluation of the artifact, and its results, are then presented and discussed. From the evaluations, we find that the limitations of the data set causes large errors between the RBF-generated data and the ground truth data. As well as this, it is found that the RBF fits the data points years into the past better than recent data points.

Contents

Chapter 1

Introduction

Visualizing data in high detail has many real-world uses, such as making accurate scientific observations, or having a more realistic view of what is going on. In this project, we aim to visualize low-resolution data in high detail, so that people can have a detailed view of how this data looks in 3D. The goal of this visualization is to be convincingly realistic, but not necessarily highly accurate.

The research problem we are faced with is that the data to be visualized is of very low detail (resolution). The data contains the heights of the surface of Antarctica at points uniformly sampled along the Earth's surface over forty million years. There are about 40,000 time slices of the data to be visualized. The low-resolution problem arises from the heights being sampled, at best, every 80km across the surface of the Earth. This means that important details, such as mountain ranges, will completely disappear between sampled points. The reason the data is of such low resolution is due to the time consumption of running the geological simulation to produce a higher resolution output for 40,000 time slices. As well as this, it becomes increasingly difficult to estimate data points years into the past as the resolution increases, due to the limitations of the geological approximations.

The main area of research which attempts to overcome this problem of low-resolution data is surface reconstruction (SR). SR approximates a surface to fit a point cloud, either through interpolation, approximation, or learning-based techniques [11]. Points along the reconstructed surface can then be added to the dataset, making it higher-resolution. SR, when applied to our data set, will produce higher resolution data, which can be rendered in 3D more realistically.

The goal of this project is to evaluate the accuracy of these different SR techniques. To evaluate how well an SR performs, we can compare the result of the SR ran on a data time slice with a data time slice at the same time, but with a greater resolution equal to the output resolution of the SR. We can then measure the height differences between the SR points and the ground truth data.

In this report, I outline the literature review and work done regarding existing techniques that apply to our case study. Secondly, I outline the implementation of the proposed solution. Finally, the evaluation performed is shown and discussed, and a conclusion is given.

Chapter 2

Related Work

The most notable research for increasing the resolution of a point cloud is surface reconstruction (SR). SR refers to the process of extracting a surface from a set of 3D points (point cloud) which is as close as possible to the original surface the points were sampled from [11]. Points lying on this surface can then be added to the data set, thereby increasing the dataset resolution. SR can either be done through interpolation, approximation, or learning-based techniques. In this section, I will discuss the advantages and disadvantages of each.

Most of the research on SR focuses on solving the problems of non-uniformly distributed point clouds, which possibly contain noise [11] [2] [6] [4] [9] [7]. This is likely due to the reduced difficulty of reconstructing a uniformly distributed point cloud, hence making it a less significant research problem. It may also be due to most point cloud data coming from 3D scans, which tend to produce non-uniformly sampled, noisy point clouds [2] [7] [11]. In our case, the data is uniformly sampled, and we assume that no noise exists in the data, due to the data arising from a verified simulation.

Regardless of the difference in focus, SR literature can still prove useful in our case study. Interpolation techniques, also known as combinatorial techniques, utilize Voronoi Diagrams and its dual Delaunay Triangulations to represent the point clouds [11]. These work well for uniformly sampled point clouds which are topologically consistent (this is true for our case, due to the surface of Antarctica containing no holes) [11].

Approximation (implicit) techniques reconstruct the surface using an implicit function [11] [3]. One drawback to this approach is the loss of geometric information at undersampled parts of the point cloud. This would be very problematic in our case, due to the low sampling resolution. Another drawback is the tendency of the function to over smooth the entire point cloud. Antarctica does not seem very smooth, so this smoothing would be an issue.

Lastly, learning-based techniques can be used to reconstruct a surface, either parametrically, or implicitly [11]. They work by performing machine learning on the point clouds to reconstruct a surface.

An example of a parametric learning-based SR is AtlasNet, introduced by [5]. This approach takes as input a point cloud, and a 2D image of the object, and learns the 3D surface by stitching together a set of 2D square patches obtained from a neural network. Although the stitching of squares fails to overlap properly, this problem was overcome by the use of 3D sphere parameterization [11]. However, this approach is still limited to closed surfaces only.

Another parametric learning-based approach by [10] uses an approximation known as the Wasserstein distance to overfit a set of deep ReLU networks, and obtain local parametrizations of subdivided input point clouds.

DeepSDF, introduced by [8], is an implicit learning-based approach that regresses a continuous signed distance function from a point cloud, with an auto-decoder structure. The approach is able to generate a water-tight surface, but fails for large-scale point clouds.

Chapter 3

Methodology

Approximation techniques approximate the surface without passing directly through the points. Because our data is from a simulation, and has been verified as highly accurate, we are treating the data points as ground truth, so we want our surface reconstruction to pass directly through the points. Learning-based techniques also do not pass directly through the points, so would not be ideal. Therefore, interpolation techniques would be much better suited.

The most common interpolation techniques for surface reconstruction rely on Delaunay Triangulations to represent a point cloud. The issue with this is that all points added to the point cloud will be linearly interpolated, which would produce flat areas between the data points. This is not very realistic, as points on the surface of Antarctica are not likely to follow a perfectly straight linear interpolation.

The solution we arrived at was to use the approximation technique called the "Radial Basis Function" (RBF), but make it an interpolation technique by forcing it to pass through the data points. This allows the surface reconstruction to pass exactly through the data points, which preserves the point cloud geometry, and to interpolate points in a non-linear way which is likely to be more realistic than a linear interpolation.

For evaluating the accuracy of the RBF, different kernel functions were tested. It was hypothesized that non-linear kernels would generate more accurate data than the linear kernel, but all of the kernels supplied by the Python programming language were tested for comparison due to the low effort of evaluating each kernel. As well as testing different kernels, a few different time slices were chosen. The first, last, and middle time slices were chosen, so that we could gain an idea of how the RBF performed for time slices throughout the data set.

For evaluating the errors, the differences in heights at each point were summed up and averaged. Although this gave a good indication of the overall error, a view of where each error was in the data would be beneficial for gaining more insight into the accuracy of the RBF. Therefore, each individual error was plotted as a red color in the location where the error was being measured. The greater the error, the greater the intensity of the red color. This gave a succinct view of what the errors looked like in each part of the data, and how well the RBF was fitting each data point.

Chapter 4

Implementation

The first part of the implementation is to understand the data we are given. The data of the heights of Antarctica, given from the Victoria University of Wellington Antarctic Research Centre (ARC), comes from a geological simulation, and is provided as a .netcdf file. The relevant information in this file is the x, y, and h attributes. The x and y attributes describe the discrete vertical and horizontal axes of the data points (in polar stereographic kilometers). The resolution of the x and y attributes is one point every 80km. The h attribute describes the height of the point above sea level (in meters).

In order to visualize the data in 3D, we must first load the data, and convert the polar stereographical coordinates into euclidean spherical coordinates, so that we can visualize the data points relative to the spherical earth. The Python programming language is a good choice for these tasks, as it is simple and fast to write, and has a large range of libraries. For reading the NETCDF file, the "netCDF4" Python library was used. For converting the data to Spherical coordinates, the "pyproj" Python library was used. The polar stereographical 2D coordinates (x and y) are first converted to latitude and longitude, and then converted to spherical coordinates lying on the unit sphere. The coordinates are then scaled by the radius of the earth (in m). Then, each point is displaced by the unit normal to the surface of the earth at the specific 3D point, multiplied by the height value.

After converting the data to euclidean 3D spatial coordinates, they need to be written to a file for 3D rendering. The OBJ file format was chosen for its simplicity and popularity. At first, the 3D x, y, z coordinates were written to the file to be visualized as points, without any faces. The software MeshLab was chosen to view the obj file as it is quick and easy to use.

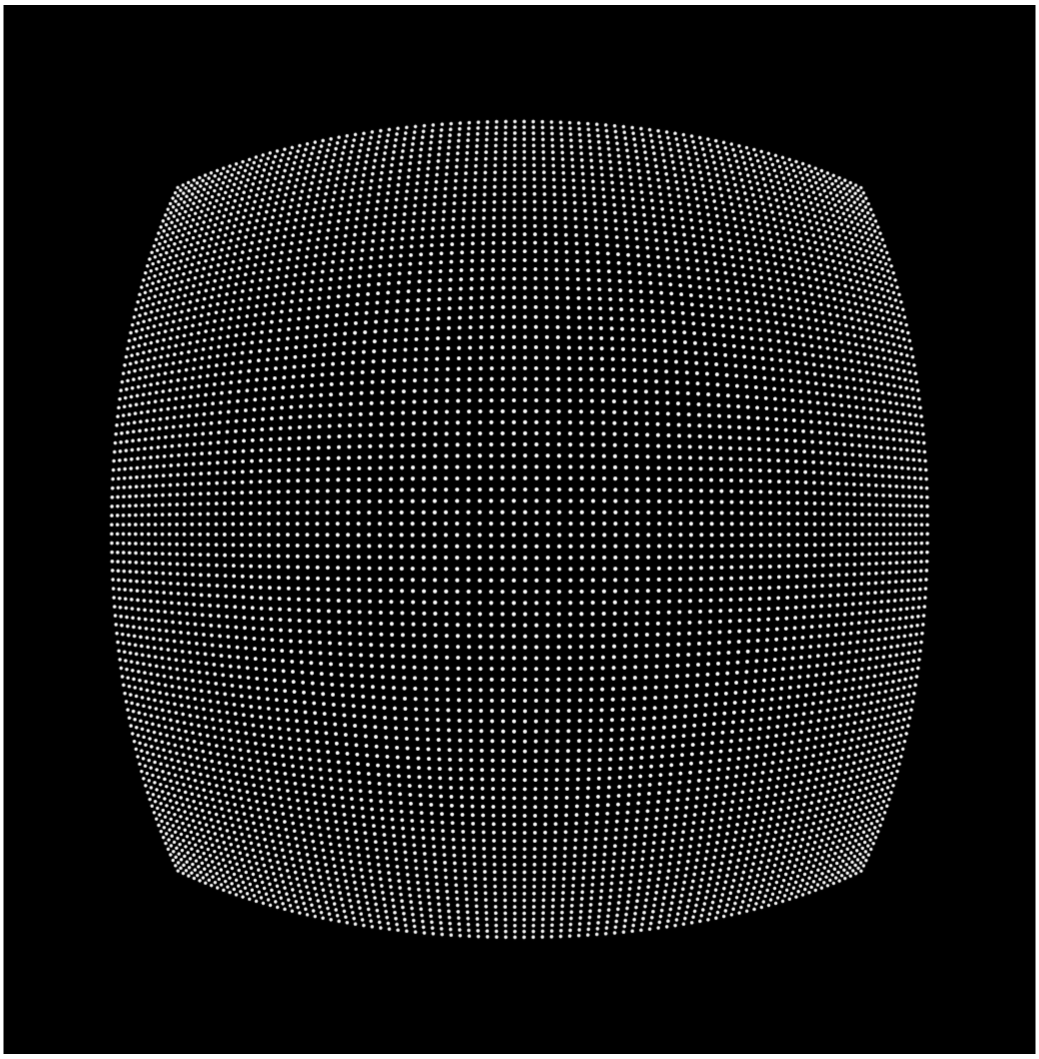


Figure 4.1: 3D point rendering of the data at t=0

To enhance the visualization, all points which lie below sea level were colored blue, and all points above sea level were colored white. This allowed a more realistic view of Antarctica, and made the boundary of ice visible.

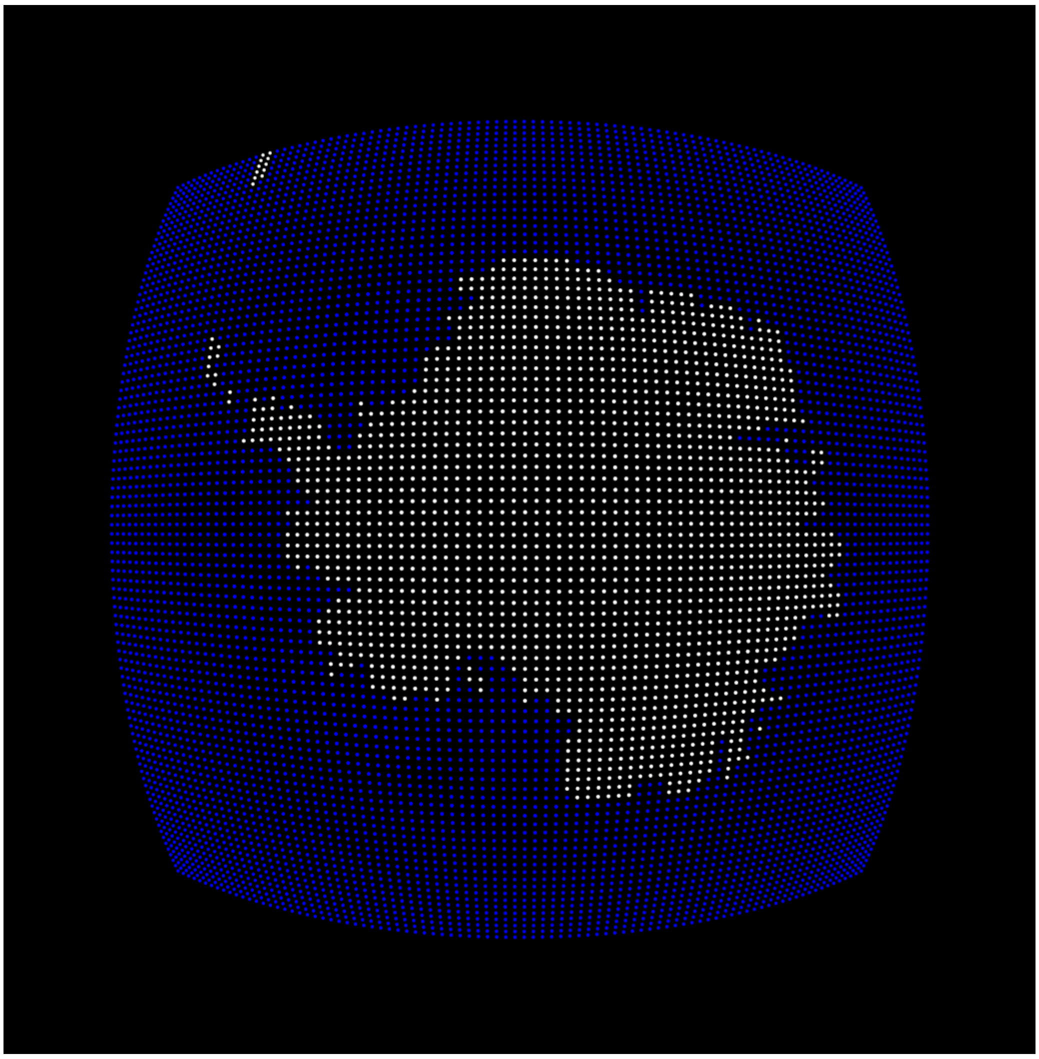


Figure 4.2: 3D point rendering of the data at t=0. White points are above sea level. Blue points are at or below sea level.

Work was also done to triangulate the point cloud, so that it could be visualized as a more realistic 3D mesh. The triangulation algorithm for our data, which lies on a uniform grid, is very simple. The OBJ file format uses face indices to render the faces, so the triangulation algorithm was written in Python to produce these indices from the point cloud produced from our data. Each triangle is expressed in the OBJ file as the 3 indices of the triangle vertices in anti-clockwise order. The vertex indices are ordered as shown in 4.3.

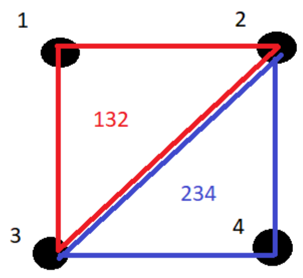


Figure 4.3: OBJ triangulation example

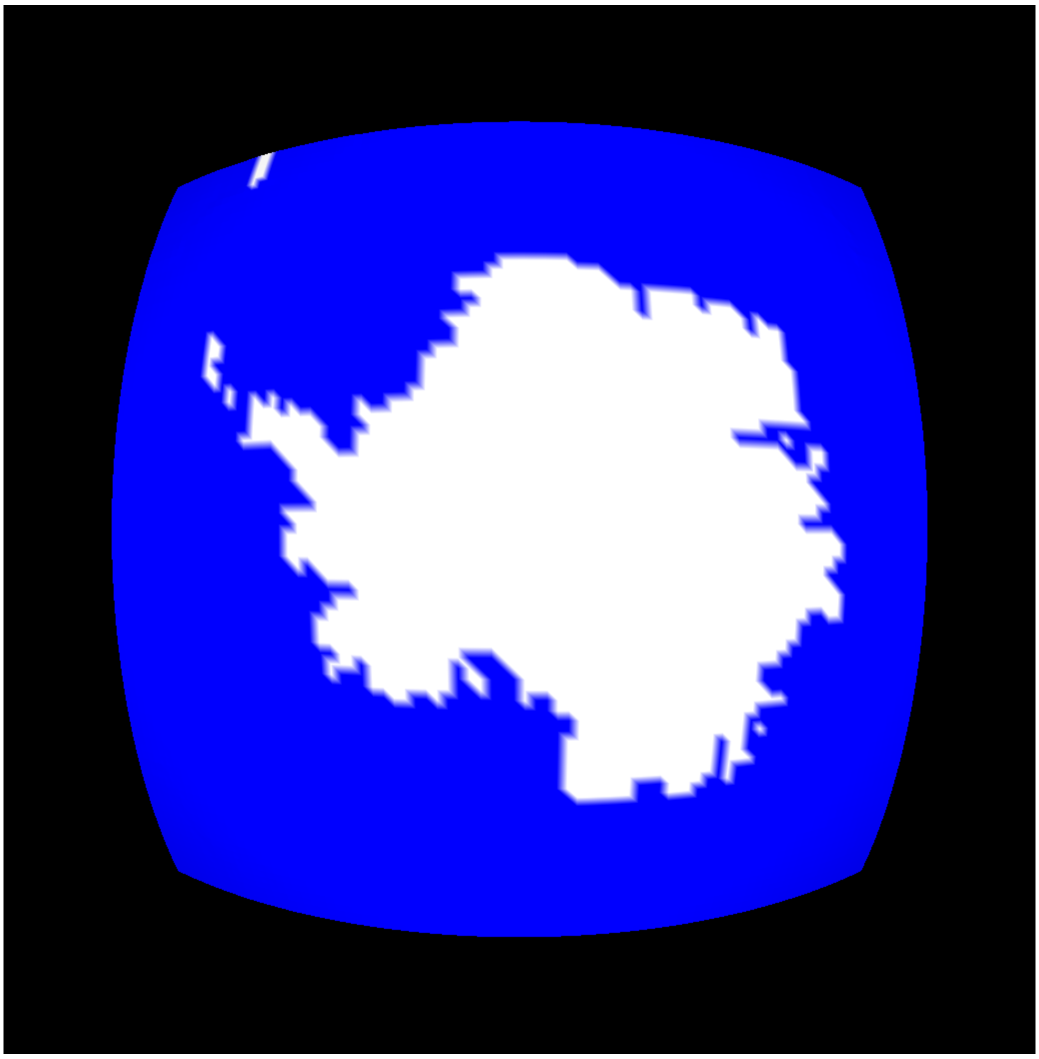


Figure 4.4: 3D triangulated mesh rendering of 4.2

For the SR, I started off with a technique that fits the data perfectly (i.e. an interpolation technique). Interpolation SR is likely to give the best results, as we treat the data points as ground truth. With this in mind, an approximation technique called the "Radial Basis Function" (RBF) [3] can be used, but turned into an interpolation technique by forcing the function to pass through every point. The RBF was ran on 4.4 using the Python library "scipy.interpolate.RBF" [1]. The RBF works by taking as input our data of the x and y polar stereographic coordinates of each point, and the height at each point. A polynomial curve (surface), which passes through the data points exactly, is then formed via interpolation. The interpolated surface's shape is determined by a "kernel function". The supported kernel functions from scipy.interpolate.RBF were used [1]. Once the surface is formed, we can sample points on the surface, and add them into our data set to make the point cloud denser. With our new higher resolution data set, we can convert the polar stereographic coordinates to Euclidean coordinates as described above. The number of points we add will be determined by the "sample density", which specifies the rate of sampling points in the x and y direction. For example, a sample density of 2 means that we sample the RBF surface at twice the resolution of the input data set, and so the final data set would be twice as large.

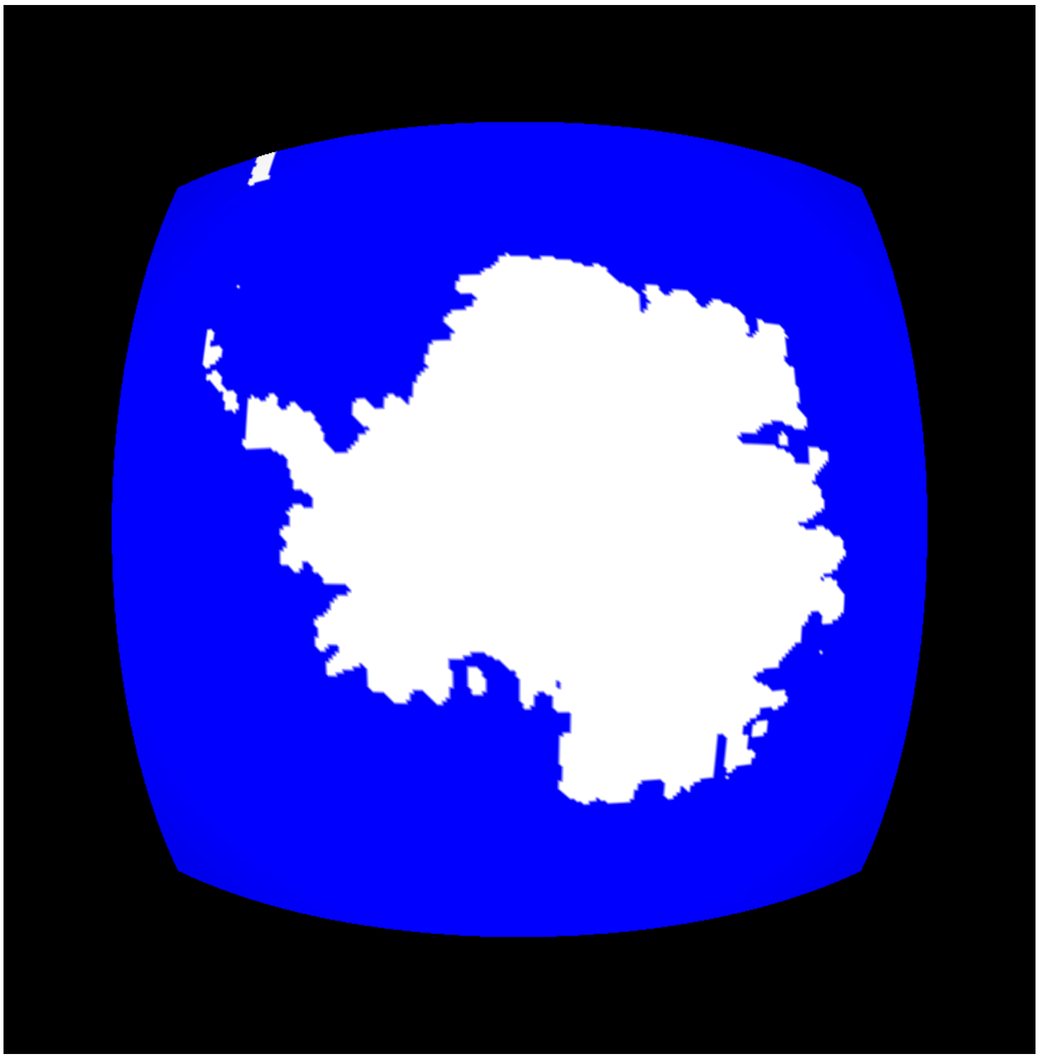


Figure 4.5: RBF of figure 4.4 using a "multiquadric" kernel, and a sample density of 3.

Chapter 5

Results

Although our aim is to increase the resolution of the data such that it looks far more realistic, measuring the value of realism in the data is difficult to do. Therefore, this evaluation will focus on numerical differences between the generated data with higher resolution and the original (ground truth) data.

The maximum resolution of the data is 80km. Therefore, if the data generated from SR were to exceed 80km, we would have no way of numerically verifying how correct it is, as we have no data beyond 80km resolution to compare it to. Therefore, the evaluations done on the SRs will compare the ground truth data with an SR generated from a lower resolution version of the ground truth data. The lower resolution ground truth data will be formed by removing every *nth* point in the x and y direction, so that there is a greater distance between each point. In the following evaluations, a time step of "t" years into the past is chosen, and the 80km resolution ground truth data is compared to the RBF data generated from the ground truth data of resolution "Starting resolution". The sample density for the RBF data is chosen such that the final resolution matches that of the ground truth data we are comparing to (80km). For calculating the "Error" (average error), we compare the data points in the ground truth data with the RBF generated data which have the same (x, y) coordinate. The Error is calculated by summing the differences in height at each two data points we are comparing, and dividing the final sum by the total number of data points.

t = 0

Evaluation	RBF kernel	Starting resolution	Error (6sf)	Starting resolution	Error (6sf)
1	multiquadric	160km	1606.78m	240km	2888.19m
2	inverse	160km	1604.98m	240km	2884.36m
3	gaussian	160km	1608.87m	240km	2891.93m
4	linear	160km	1597.14m	240km	2871.53m
5	cubic	160km	1606.01m	240km	2886.63m
6	quintic	160km	1607.53m	240km	2890.35m
7	thin plate	160km	1603.95m	240km	2882.46m

t = 20,000,000

Evaluation	RBF kernel	Starting resolution	Error (6sf)	Starting resolution	Error (6sf)
8	multiquadric	160km	1567.79m	240km	2793.58m
9	inverse	160km	1566.15m	240km	2791.12m
10	gaussian	160km	1569.96m	240km	2797.58m
11	linear	160km	1558.87m	240km	2781.59m
12	cubic	160km	1567.07m	240km	2792.66m
13	quintic	160km	1568.50m	240km	2794.82m
14	thin plate	160km	1565.23m	240km	2790.04m

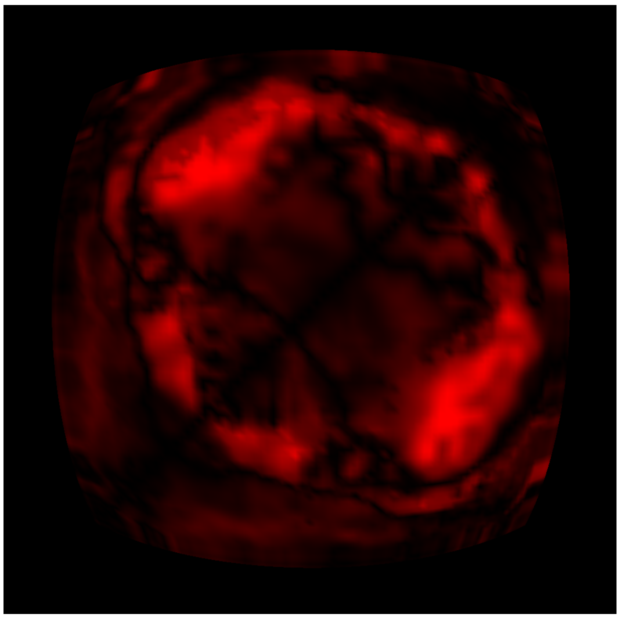
t = 40,000,000

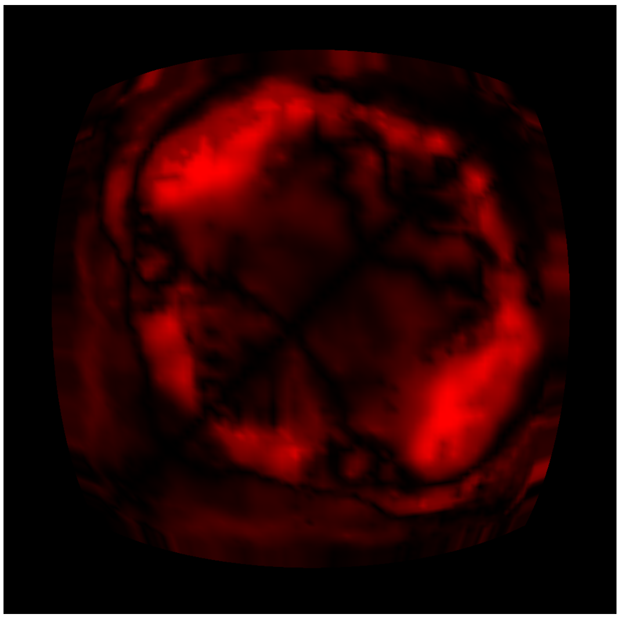
Evaluation	RBF kernel	Starting resolution	Error (6sf)	Starting resolution	Error (6sf)
15	multiquadric	160km	1379.22m	240km	2403.62m
16	inverse	160km	1377.77m	240km	2401.11m
17	gaussian	160km	1381.42m	240km	2407.83m
18	linear	160km	1371.39m	240km	2391.39m
19	cubic	160km	1378.55m	240km	2402.53m
20	quintic	160km	1379.86m	240km	2405.03m
21	thin plate	160km	1376.90m	240km	2399.79m

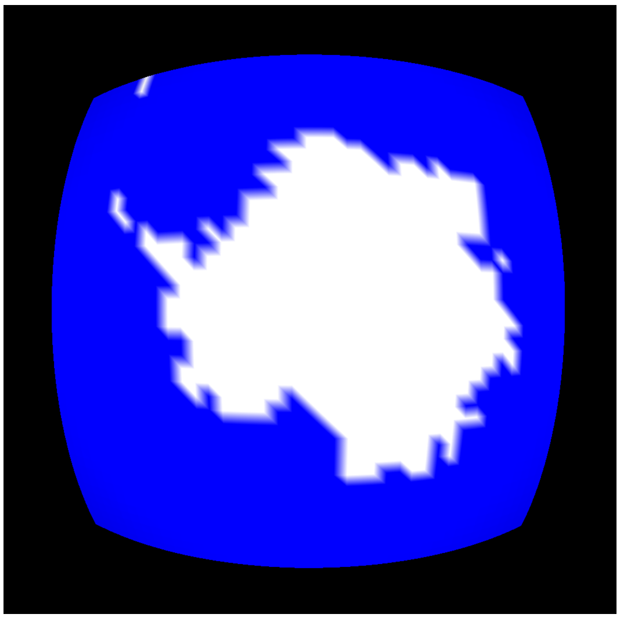
I have also plotted "Error plots" using the "Error" of each point (from the evaluation) in the RBF-generated data set. The minimum and maximum error (min error and max error) of all of the height differences in the RBF generated data of a single evaluation and starting resolution is taken. The range of the error is mapped from [min error, max error] to [0, 1], and is used as the red component (r) in the point color (r, 0, 0). Error plots are shown for evaluations 3, 4, 17, and 18 - the highest and lowest Error RBFs (gaussian and linear respectively) - with a starting resolution = 160km.

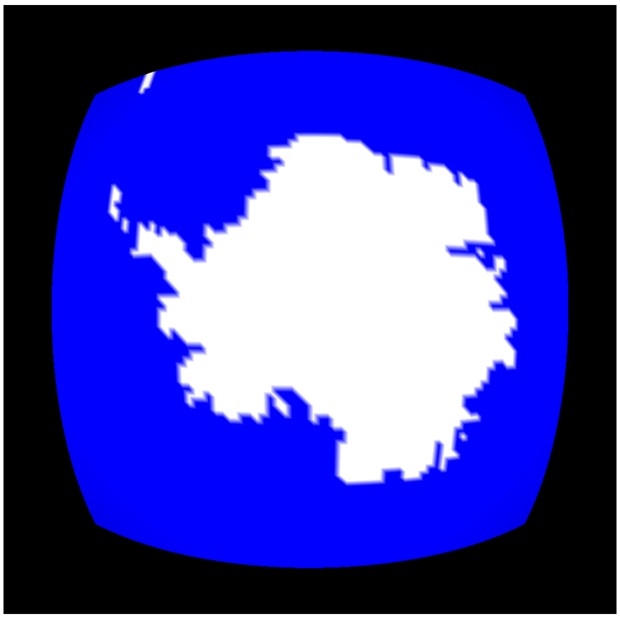
The 3D triangulated data of these evaluations are also shown shortly after, alongside the ground truth triangulation for resolutions 160km and 80km respectively.

t = 0


(a) RBF kernel = Guassian


(b) RBF kernel = Linear

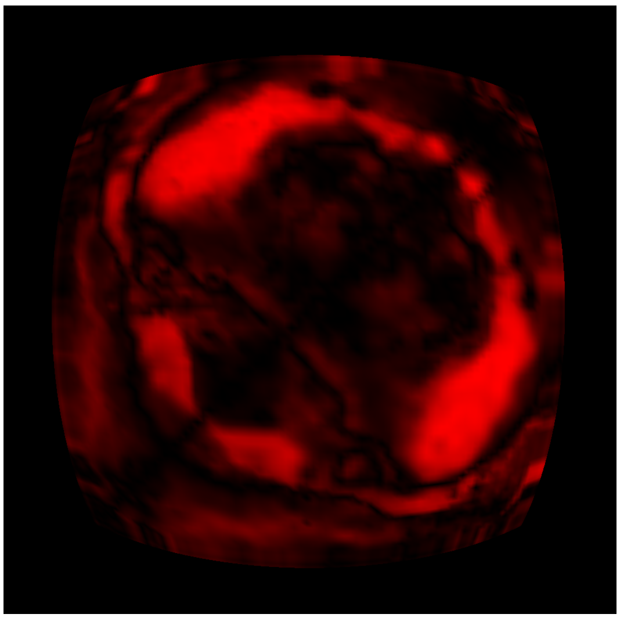

(c) Ground truth data 160km resolution

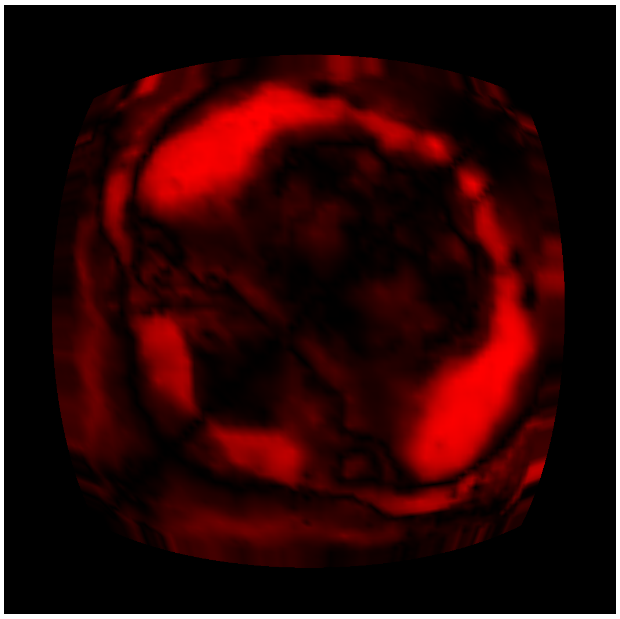

(d) Ground truth data 80km resolution

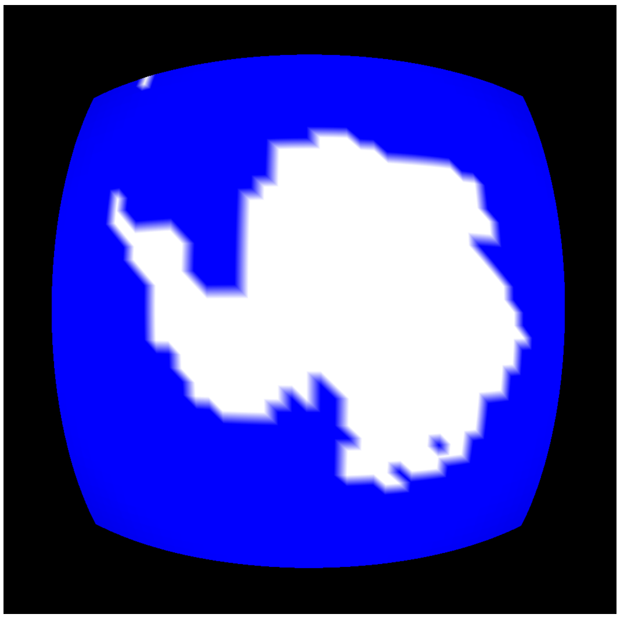
(e) RBF kernel = Guassian

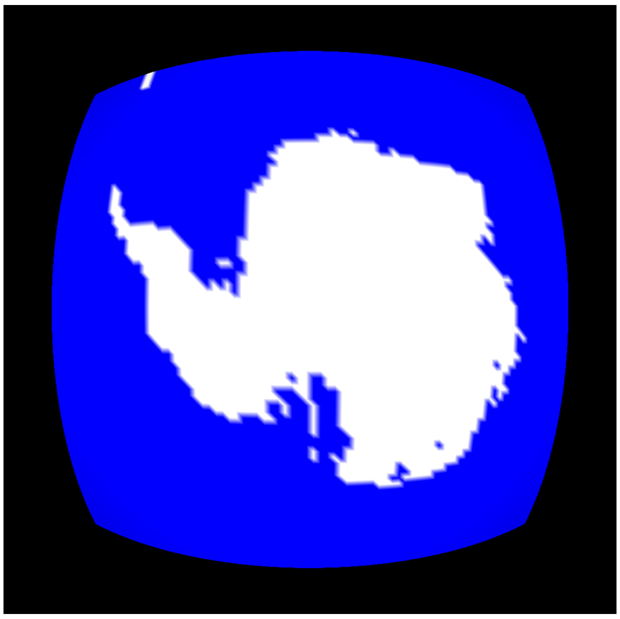
(f) RBF kernel = Linear

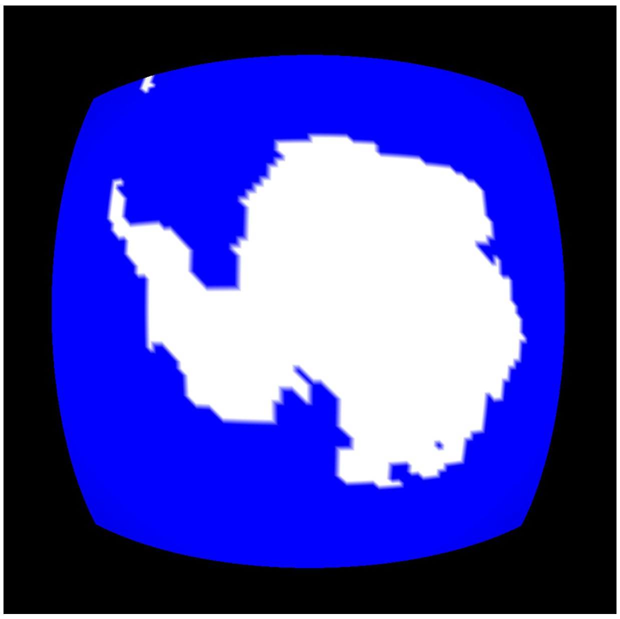
t = 40,000,000

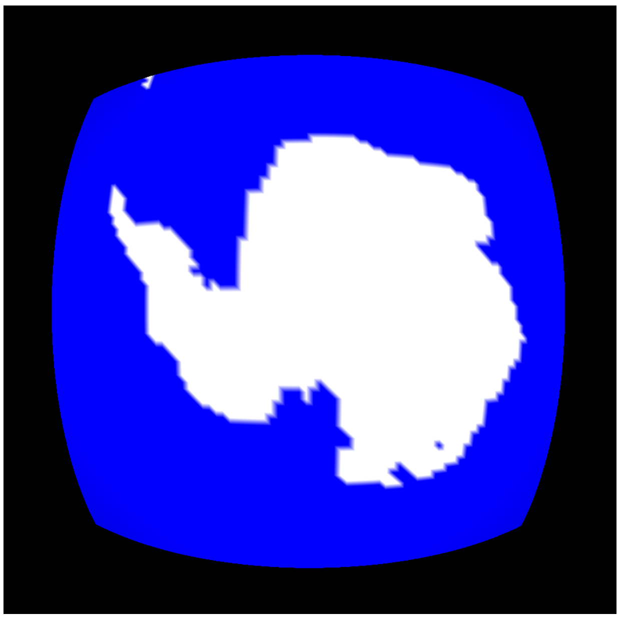

(g) RBF kernel = Guassian


(h) RBF kernel = Linear


(i) Ground truth data 160km resolution


(j) Ground truth data 80km resolution


(k) RBF kernel = Guassian


(l) RBF kernel = Linear

Chapter 6

Discussion

From the data collected, the Linear RBF kernel has the lowest error and the Gaussian RBF kernel has the highest error in all evaluations. This disproves my initial hypothesis that a non-linear RBF kernel would give a lower error, due to being able to better model the Antarctic ice heights, which intuitively would be more likely to follow a non-linear pattern.

Although the Linear RBF kernel does have the lowest error, it is only by a small margin. Comparing evaluations 3 and 4 with a starting resolution of 160km, the linear kernel only gives approximately a 0.7291% (4sf) improvement (percentage change) over the gaussian kernel.

It is clear from the evaluation tables that all RBF kernels have a similar error. We can also see from the error plots that both the linear and gaussian RBFs give higher error values in the areas where the boundary between ice and ocean occurs. This could be due to the rapid change in height values as the ocean boundary meets the ice shelf, which would mean that a linear interpolation of points would be an interpolation across a greater height difference. In this case, the ground truth data point is a lot more likely to lie a great distance from the interpolated data point.

Another reason the errors are substantial could be because we are starting with a very low resolution of 160km. This resolution is very limited in accurately describing the shape of the surface of Antarctica, and we do not know what kind of height variations could be causing the ground truth data points to be placed where they are. Because we are interpolating across a large resolution, we are bound to have large errors, as we do not have any additional information on how the surface of Antarctica behaves.

Another interesting observation is that the ground truth data of 80km resolution at t = 0 shows a spec of ice below and to the left of the image center. This spec of ice is not present in the ground truth data of 160km. Because of this, the RBF had no knowledge of this spec of ice, so was unable to generate it. This highlights the limitation that there may be significant geometric details left out of the data because of the low resolution, and these details can not always be generated by the RBF.

For evaluations 3 and 4, where the starting resolution is 240km, we get very similar results for evaluations 3 and 4 with a starting resolution of 160km; only all of the errors are increased by about 1280m. The increased error is very much what we would expect, as the input data is of lower resolution, so the RBF has to interpolate over larger distances between the point's (x, y) coordinates, leading to a greater chance of being far away from the ground truth data point. Interestingly, we see again that the linear kernel has the lowest error, and the gaussian has the highest. This indicates that the input resolution does not affect the quality of the RBF kernel very much. This could indicate that the RBF kernel quality is determined more by the geometric shape of the data, rather than the resolution.

For a starting resolution of 160km, evaluations 17 and 18, where t = 40,000,000, have errors that are smaller than evaluations 3 and 4 by about 220m. This may be due to evaluations 17 and 18 having data that is a lot flatter in most areas. The error plot for evaluations 17 and 18 is a lot more contrast than that of evaluations 3 and 4, indicating that the data varies less overall.

Chapter 7

Conclusion

In order to render the data set of Antarctic heights with greater visual accuracy, an artifact was developed. The main goal of this artifact was to increase the resolution of the data in such a way that the higher resolution data was visually accurate and realistic as possible.

Because the data was verified as being highly accurate, we could treat the data points as ground truth when creating and evaluating a higher resolution data set. Given that we are treating the input data as ground truth, it was decided that the techniques reviewed which were most likely to give the highest accuracy were interpolation techniques.

Given that interpolation techniques appeared to be the best option, the RBF interpolation technique was chosen. From this, an artifact was developed, which used the RBF to generate data points at a higher resolution. The accuracy of the newly generated data was evaluated for different RBF kernels, time slices, and data resolutions. Overall, it was found that all RBF kernels have a similar, significantly large error. Interestingly, the RBF fits the older data time slices much better than the more recent ones.

The most significant limitation to the accuracy of the higher resolution generated data is the low resolution of the input data set. Without any additional information, it is difficult for SR techniques to accurately generate data that is close to the ground truth.

Chapter 8

Future Work

After learning of a higher resolution present-day data set called “Reference Elevation Model of Antarctica” (REMA), it is thought possible to use this data to aid in the development of the artifact. The REMA data set contains the heights of Antarctica’s surface, similar to the current data set used for this project (model data). The data is informed through satellite imagery, which provides much higher detail than the simulation-generated model data used for this project. The highest resolution possible in the REMA data set is 8m, which is far greater than the highest resolution of the model data (80km).

The REMA data could be used as a 3D visualization for the present day, but could also inform the model data at time slices that span millions of years into the past. If the changes in the model data at each time slice were applied to the REMA data, a high resolution, accurate 3D visualization could be created which closely resembles the geometry of the model data. This method would provide far greater accuracy than using SR techniques on the limited low resolution 80km model data set.

Bibliography

[1] scipy.interpolate.rbf. https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.htm 2021.

[2] BERGER, M., TAGLIASACCHI, A., SEVERSKY, L. M., ALLIEZ, P., GUENNEBAUD, G., LEVINE, J. A., SHARF, A., AND SILVA, C. T. A survey of surface reconstruction from point clouds. *Computer Graphics Forum 36*, 1 (mar 2016), 301–329.

[3] CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MC-CALLUM, B. C., AND EVANS, T. R. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01* (2001), ACM Press.

[4] FLEISHMAN, S., COHEN-OR, D., AND SILVA, C. T. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics 24*, 3 (jul 2005), 544–552.

[5] GROUEIX, T., FISHER, M., KIM, V. G., RUSSELL, B. C., AND AUBRY, M. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR abs/1802.05384* (2018).

[6] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, Association for Computing Machinery, p. 71–78.

[7] MOREL, J., BAC, A., AND VEGA, C. Terrain model reconstruction from terrestrial LiDAR data using radial basis functions. *IEEE Computer Graphics and Applications 37*, 5 (2017), 72–84.

[8] PARK, J. J., FLORENCE, P., STRAUB, J., NEWCOMBE, R. A., AND LOVEGROVE, S. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR abs/1901.05103* (2019).

[9] WANG, W., POTTMANN, H., AND LIU, Y. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics 25*, 2 (apr 2006), 214–238.

[10] WILLIAMS, F., SCHNEIDER, T., SILVA, C., ZORIN, D., BRUNA, J., AND PANOZZO, D. Deep geometric prior for surface reconstruction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (jun 2019), IEEE.

[11] YOU, C. C., LIM, S. P., LIM, S. C., TAN, J. S., LEE, C. K., AND KHAW, Y. M. J. A survey on surface reconstruction techniques for structured and unstructured data. In *2020 IEEE Conference on Open Systems (ICOS)* (nov 2020), IEEE.