EMODnet Thematic Lot n°0 –*Bathymetry –High Resolution Seabed Mapping (HRSM2)*

EASME/EMFF/2017/1.3.1.2/01/SI2.791269

WP4: Web Visualization, Terrain Tiling and Interpolation Methods

Technical report

Date: 28/12/2020

Prepared by: Ricard Campos and Josep Quintana (Coronis)

Table of Contents

1. Introduction

The report describes the activities undertaken by Coronis within the EMODnet HRSM 2 project.

2. Web Visualization and Terrain Tiling

During the EMODnet HRSM 2 project, Coronis has continued the work on converting raster DTMs, as the ones generated within this project, into triangulated irregular networks (TINs) optimized for web-based visualization. This structure follows a pyramid of TINs representing distinct levels of detail (LOD), where each level of detail is composed of small tiles of a fixed size, as shown in Figure 1.

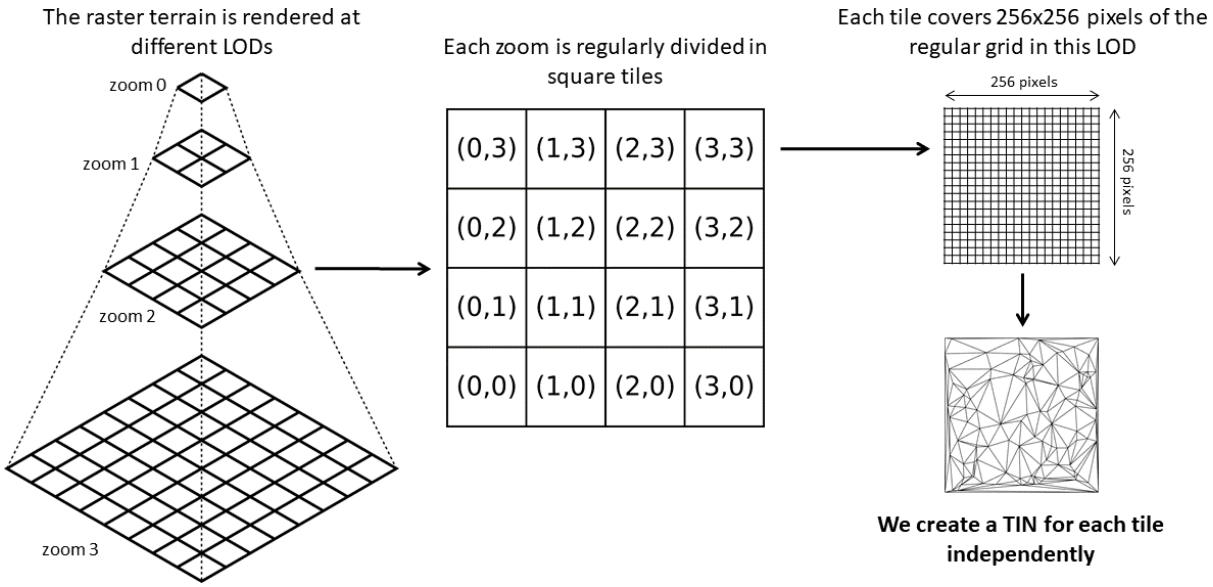

Figure 1. *Overview of the tiling process.*

In addition to refactoring code and solving some bugs that were not detected on the first version, the contributions to the state of the art developed during the first EMODnet HRSM project were published in a high-impact open-access journal:

Campos, R., Quintana, J., Garcia, R., Schmitt, T., Spoelstra, G., & M. A. Schaap, D. (2020). 3D Simplification Methods and Large Scale Terrain Tiling. *Remote Sensing, 12*(3). Available at: https://www.mdpi.com/2072-4292/12/3/437/htm

Our main contribution described in that article focused on how to create these TINs per tile, which allows us to parallelize the processing, while restricting the vertices at the borders of each tile to coincide within the same level of detail, so as to create a continuous mesh. In this paper we describe the modifications performed to three different TIN creation algorithms (greedy insertion, edge-collapse simplification, and point set simplification) to adhere to these restrictions, and we compare their results, both qualitatively and quantitatively, using the global 2018 EMODnet DTM as input.

Finally, we applied the updated code to generate the 3D visualization of the final global DTM resulting from this project.

3. Interpolation Methods

Coronis also focused on developing interpolation techniques that could help in processing the data collected during this project, as an alternative to the solution originally implemented in Globe, and that could potentially outperform the results generated by such.

We started by performing a review of the interpolation methods implemented in state-of-the-art open-source and commercial software solutions, so as to detect the most versatile solution that could fit all our needs. More precisely, we reviewed the following solutions:

- ArcGIS (ESRI)
- MapInfo Pro Advanced (Pitney Bowes)
- Maptitude (Caliper)
- Surfer (Golden Software)
- QGIS
- GRASS (Open Source Geospatial Foundation - OSGeo)
- SAGA GIS

From this review, we identified the following different methods:

- Nearest Neighbors
- Delaunay (linear) [Delaunay1934]
- Natural Neighbors [Sibson1981]
- Inverse Distance Weighted [Shepard1968]
- Radial Basis Functions [Fasshauer2007]
- Kriging [Williams1998]
- Moving Least Squares [Levin1998]

We performed a review of the theoretical and practical aspects of each method. We implemented all of these methods in Matlab in the **heightmap interpolation toolbox**, and released the code publicly with an open-source GPLv3 license at the following link:

https://github.com/coronis-computing/heightmap_interpolation_toolbox

While Matlab is not suitable for performant processing of large scale data, this toolbox was developed as a reference implementation of the methods, so that we could test their suitability for our needs.

Figure 2 shows a sample of the behaviour of each method on a single sample dataset.

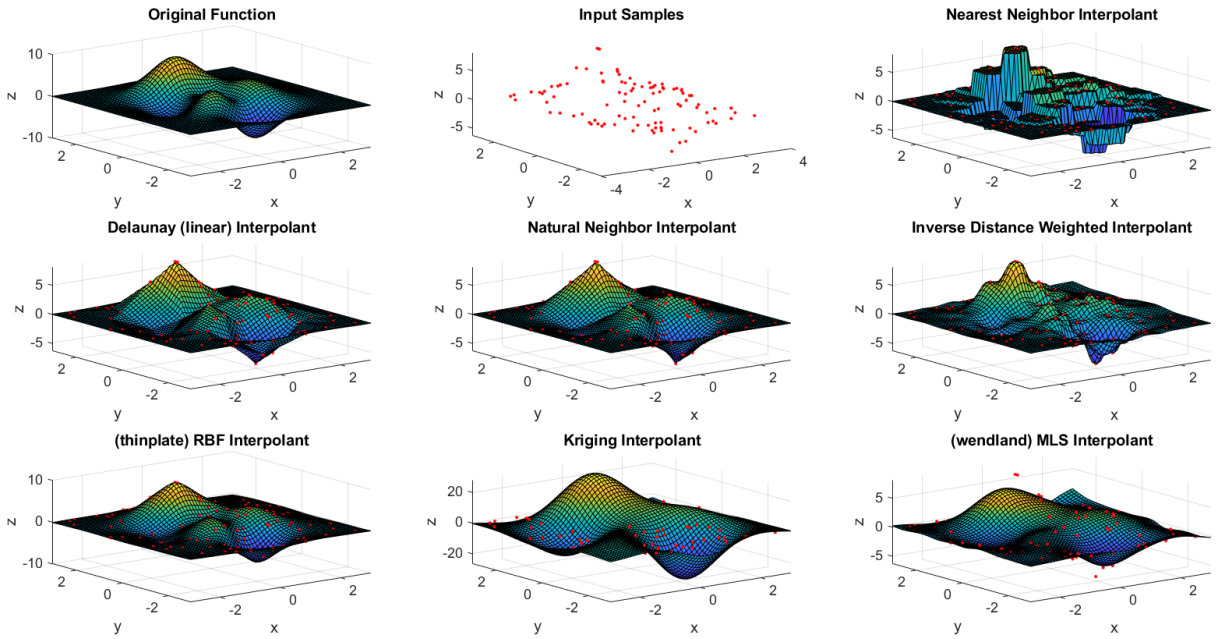

Figure 2. *Sample results for all the interpolation methods implemented in the toolbox.*

3.1 Radial Basis Functions Interpolation

After reviewing and testing the interpolation methods listed above, we found the most versatile to be the Radial Basis Function (RBF) interpolator. A RBF is a function whose value depends only on the distance between the input and some fixed point. The basic idea of a RBF interpolator is to construct an interpolant of the data using a summation of several RBF centered at the input data points. The formal definition is the following:

$$s(x) = p(x) + \sum_{i=1}^{N} \lambda_i \phi(|x - x_i|)$$

Where:

- $s(x)$ is the interpolant evaluated at point x.
- $p(x)$ is a polynomial of small degree evaluated at point *x*
- $\phi(|x - x_i|)$ is the RBF centered at a known data point x_i.
- λ_i is a scalar weight.

Thus, basically, we have a polynomial (1st term) capturing the main *trend* of the data, and the summation of weighted RBFs (2nd term). Therefore, the unknowns of this interpolant are mainly the few terms of the polynomial and the λ_i weight of each RBF. These unknowns can be solved using a linear system of equations. In matrix form, this corresponds to:

$$\begin{pmatrix} A & P \\ P^{\mathsf{T}} & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = B \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix},$$

where:

$$A_{i,j} = \phi(|x_i - x_j|), \qquad i,j = 1,\ldots,N,$$
$$P_{i,j} = p_j(x_i), \qquad i = 1,\ldots,N, \quad j = 1,\ldots,\ell.$$

While solving this system of equations is simple, it is important to notice that the matrix *A* is a square matrix with side length equal to the number of input data points. Therefore, this formulation becomes prohibitively complex for large datasets, as the amount of memory and computational resources required for solving and/or evaluating the interpolant is too large.

In order to solve this issue, we use a partition of unity (PU) approach [Wendland2002]. The PU consists of computing local RBF in small subdomains, and then blend the local contributions together using Compactly-supported RBF as weighting functions:

$$\varphi_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{j=1}^{n} w_j(\mathbf{x})}$$

$$\sum_i \varphi_i \equiv 1 \text{ on } \Omega.$$

$$f(\mathbf{x}) \approx \sum_i \varphi_i(\mathbf{x}) Q_i(\mathbf{x})$$

$$\Omega \subset \bigcup_i \text{supp}\,(w_i)$$

Where Ω is our query domain (i.e., the domain where the interpolant will be valid) and $Q_i(\mathbf{x})$ is a RBF interpolant, following the same formulation presented above. Therefore, the idea is to first divide the domain into manageable sub-domains where a RBF interpolant can be computed. Then, for a given query point, the interpolated value at that position is the result of merging the contributions of nearby RBF interpolants. Moreover, the key is to use compactly-supported RBF to weight the contribution of each subdomain. Since their support is limited, this means that at some point the contribution of each subdomain is zero, and can be disregarded. Therefore, local RBF interpolants away from the query point do not contribute at all to the final interpolation value. This is illustrated in Figure 3.

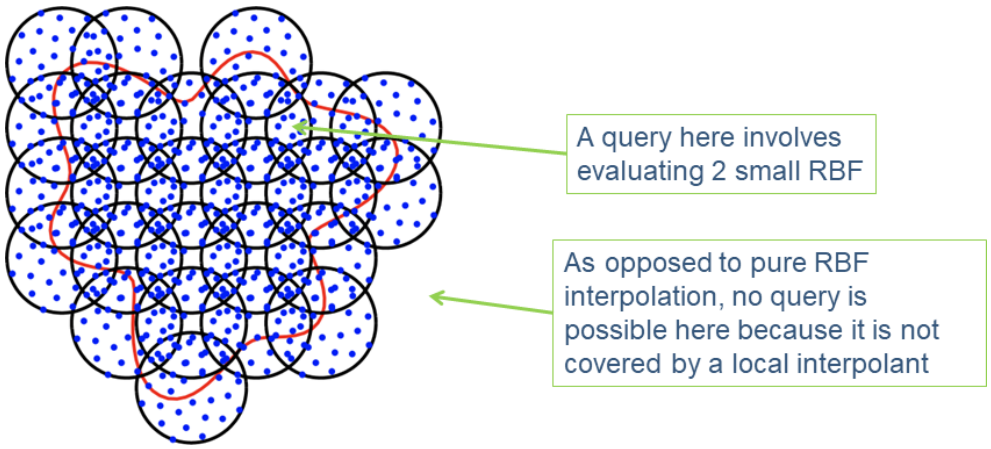

Figure 3. *The query domain, delimited with a red line, is covered by a set of overlapping sub domains, marked as black circles. The input data points, the blue dots, are contributing to one or more subdomain. Since the support of each sub-domain is limited to that of the black circle, a query point involves, at most, evaluating four nearby RBF interpolants of very few points.*

Note that the queries cannot be executed in places not covered by any local RBF domain. Original image extracted from [Larsson2017]*.*

In this way, we generated a global interpolant without requiring to use all the input points at once for its construction and evaluation, greatly reducing the computational impact of the original RBF formulation. The main issue then is how to define the sub-domains in a general way. In densely or regularly sampled datasets as the one shown in Figure 3, using domains of a fixed size is a good option. However, for datasets of variable density, finding a single domain size that fits all samples is more involved. This is the case for bathymetric datasets as the ones used in this project, since the data may come from different sensing modalities with variable resolution and/or coverage. For this reason, we define our subdomains using a circle covering following a Quadtree decomposition [Samet1984] (Figure 4).

This decomposition follows a tree data structure in which each node represents a square area. If a given node does not comply with a defined termination criteria, it is decomposed into four sub-nodes by bisecting each side of the parent square, and this process is repeated recursively. In our case, the termination criteria requests a node to contain a minimum number of points within the domain (so that the local RBF to be computed will be well defined) and a minimum cell size (so that the cells are not too small). However, we do not consider only the square domain of the quadtree node but a circular domain centered at the node and expanding out of the square a given percentage defined by a user parameter. This also ensures overlap between subdomains.

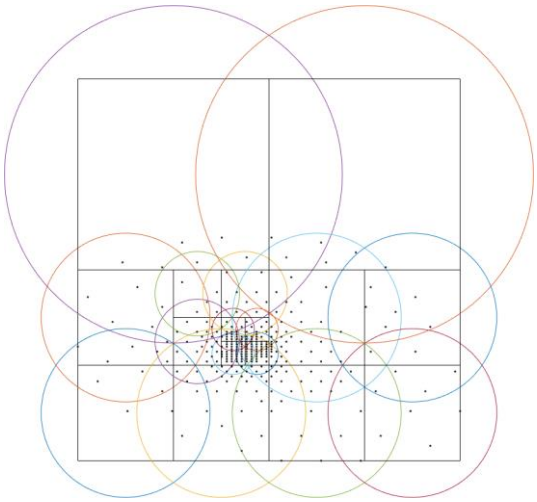

Figure 4. Quadtree circle covering on a dataset with variable density. We show the original quadtree squares and the different circular subdomains derived from them. Note how the quadtree decomposition allows defining a coverage where sparse areas are represented by larger domains, while denser areas are covered with smaller domains.

Moreover, as also pointed out in Figure 3, and due to the compact support of the local RBF interpolants, this type of interpolant is only defined on the areas covered by a subdomain. This directly means that we need to cover the desired query domain with at least one subdomain. Furthermore, each sub-domain requires covering a given number of data points so that the local RBF interpolant computed is meaningful. This is not a problem when interpolating small gaps of missing data in a dense dataset, but trying to interpolate large areas of missing data in a dataset of varying sampling density such as the one depicted in Figure 5 may not be

possible. Indeed, following the decomposition presented above will lead to sub-domains without input points, where we cannot compute a valid local interpolant. The only solution, as shown in Figure 5, would be to iteratively increase the coverage of empty subdomains until they contain a minimum number of points so as to compute a valid RBF interpolant. However, this would increase redundancy in those areas, as many local RBF interpolants would contribute to the same area, and the advantage of evaluating a small number of local interpolants from the PU would be lost.

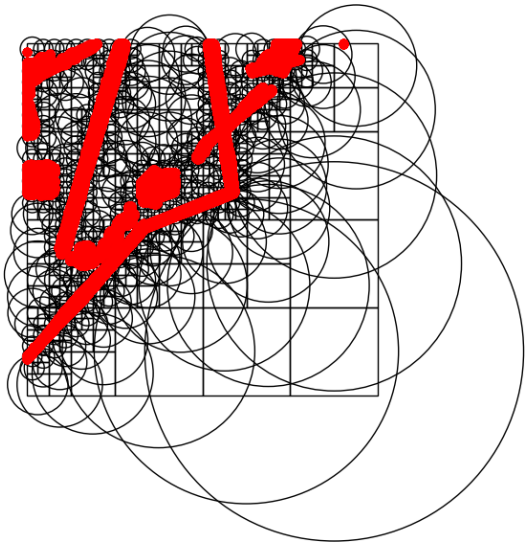

Figure 5. *Using the Quadtree PU and then increasing the subdomains until a minimum number of points falls within them leads to redundancy, as many subdomains cover the same area with no data, increasing the computational cost of evaluating the interpolant.*

To solve this issue, we purge from the quadtree those nodes/subdomains that are redundant by following a simple heuristic: if a sub-domain is contained within a domain from a shallower node in the tree, we disregard it. An example of the subdomains resulting from this heuristic are shown in Figure 6 (c).

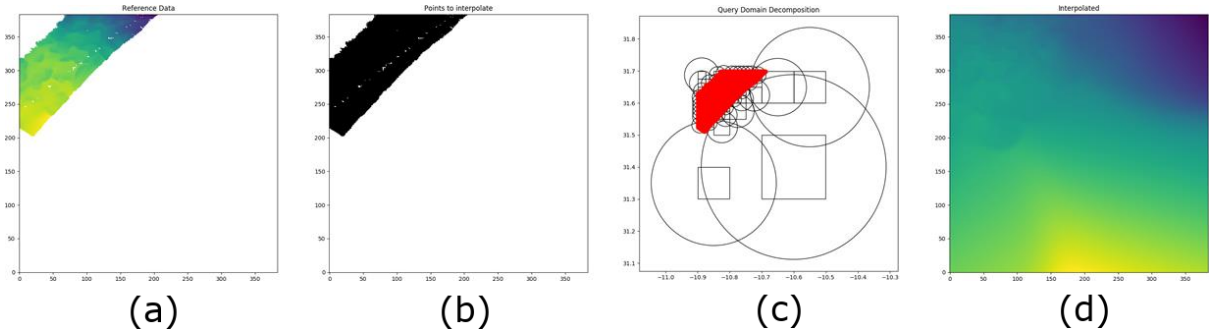

Figure 6. *The input data points with colored elevation values are shown in (a). In (b) we show in white all the points we want to interpolate. After using our quadtree decomposition and an heuristic to delete redundant sub-domains our decomposition looks as shown in (c). Finally, (d) shows the results after applying the interpolation.*

Both the pure RBF and the Quadtree PU-RBF interpolants were first implemented in Matlab, as part of the **heightmap interpolation toolbox**, and then ported to Python, to ease their implementation within IFREMER's Globe software. The Python implementation was released under LGPLv3 license at:

https://github.com/coronis-computing/heightmap_interpolation

While this new technique builds upon concepts used in many research articles on interpolation of large datasets, is not specifically following any existing solution on the reviewed related softwares, and it provides a new method that is unique to the EMODnet project.

3.2 Inpainting Interpolation

The methods reviewed in the previous section define a global interpolant that can be queried at an arbitrary point within our domain, and as such they can be used for *any* use an interpolant may be required for. For example changing the resolution of a dataset (superresolution and/or simplification), filling holes/missing data regardless of their quantity or continuity, etc.

However, during the first evaluation of the methods implemented in Globe (mid 2020), we detected that using a global interpolant may not be the best option in some cases. For instance, even if using the more efficient Quadtree PU variant, the computational complexity may be too large for filling large gaps, since we need to apply the interpolant to each of the query points. In fact, the datasets being processed in Globe already follow some gridding, and consequently the interpolant is not queried at an arbitrary point within the domain, but at predefined positions following the underlying grid of the dataset.

Therefore, within this project we took advantage of the interpolation happening on a regular grid to use *image inpainting* techniques. Inpainting methods deal with the filling of missing or deteriorated parts in an image so as to recover a complete image. Therefore, it is just another form of interpolation, and all the methods described above could be used to inpaint an image. However, image inpainting techniques also take advantage of the problem taking place on the regular lattice of the pixels in an image. Of special interest are those inpainting/interpolation methods defined as a solution to a Partial Differential Equation (PDE), because solving a PDE on a regular grid reduces to a memory-efficient iterative process using the Finite Differences method [Smith1985]. In addition, PDEs easily include additional terms such as *stiffness* or *smoothing*, which can modify the behaviour of the interpolant (see Figures 7 and 8 for a synthetic and a real-world example).

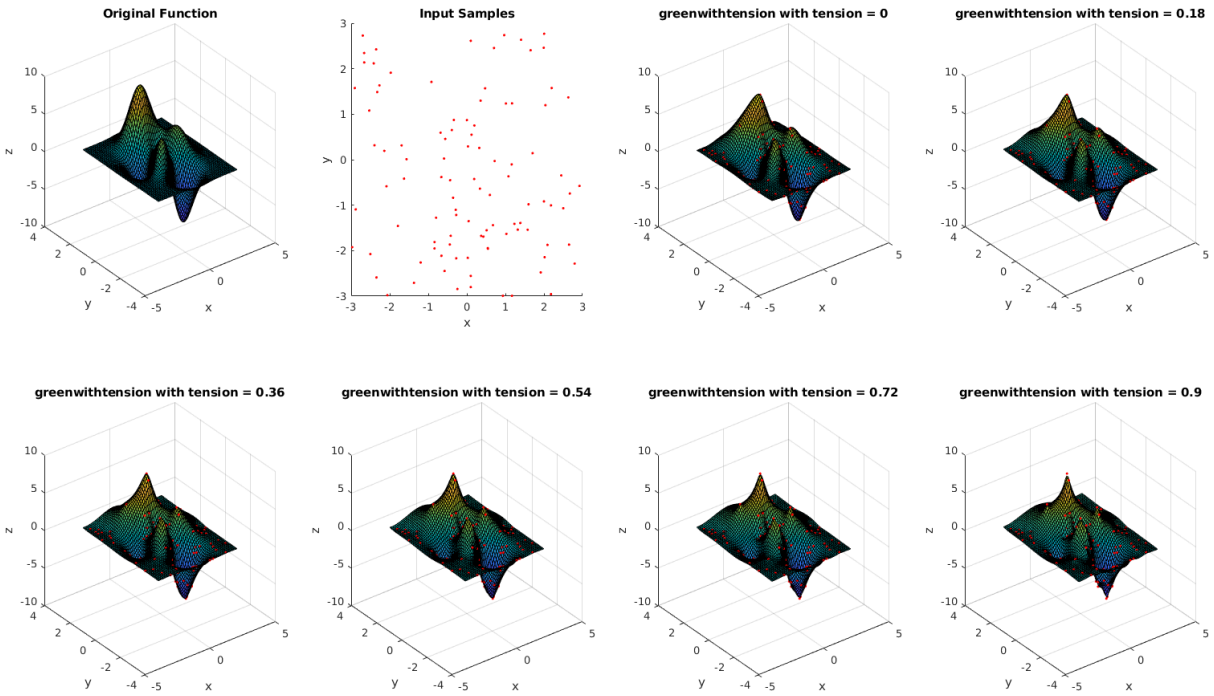

Figure 7. *Example of tuning the tension parameter of a Green spline* [Wessel2009]*. No tension results in a smooth biharmonic surface that passes through the input points. As tension increases, the interpolant changes more abruptly towards data points, resulting in more peaks towards the input data.*

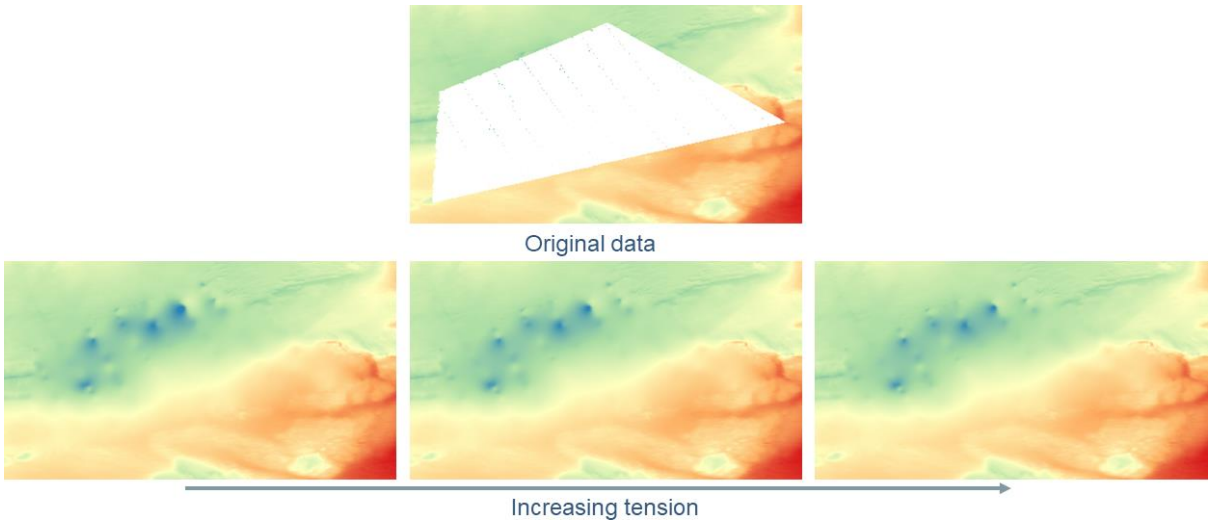

Figure 8. *Example of varying tension parameter, as in Figure 7, but this time applied to a dataset where just a few hundred samples are known in a large hole with no data (white). The effect is the same as observed before, when the tension increases, the terrain approaches the known data points with more steepness. Data courtesy of the Swedish Maritime Administration.*

After a revision on the state of the art of inpainting methods for the case of interpolation on elevation/bathymetric data, we identified some cases that were worth implementing and testing:

- **Sobolev**: fills unknown data by minimizing the Sobolev norm of the elevation values in the grid.
- **Total Variation** (TV), minimizes the total variation norm, formulated as a PDE.

- **Continuous Curvature Splines in Tension** [Smith90], the PDE equivalent of a RBF interpolation using Green functions. It allows tuning a *tension* parameter. When tension = 0, it behaves as a biharmonic interpolant, and when the tension = 1, as a harmonic interpolant, a value in between is a mix of both. Figure 8 corresponds to this method.
- **Absolutely Minimizing Lipschitz Extension** inpainter [Almansa02]. According to the authors, an inpainting method specifically designed for elevation data. A relevant property of this method is that it avoids extrapolating the data away from the measured minimum and maximum elevation values.
- **Bertalmio** inpainter [Bertalmio00]. It takes into account the gradients incident to the borders of the areas to interpolate (large areas of missing data). However, this method is specifically designed for hole filling, meaning that it does not take into account isolated points, it just fills continuous regions of data. *This last method is still under development, and the results obtained in its current implementation need to be improved.*

(a) Input data

(b) Sobolev (c) Total Variation (d) CCST (e) AMLE

Figure 9. *Example results of the inpainting methods developed. Data courtesy of the Swedish Maritime Administration.*

The inpainting methods developed were also publicly released as part of the **heightmap interpolation toolbox** described above. A sample of their results is shown in Figure 9.

While the implementation of inpainting methods within Globe is left as future work, a first internal implementation of the methods is started at the [inpainting] branch of the Python repository (https://github.com/coronis-computing/heightmap_interpolation).

4. Gridding Artifacts

During this period, Coronis tried to automatically identify gridding artifacts present in the global DTM, as a result of merging wrongly transformed maps (Figure 10). We identified the source of the gridding problem to be a problem from some softwares when converting from Cartesian UTM reference systems to the geodetic coordinates used in EMODnet products.

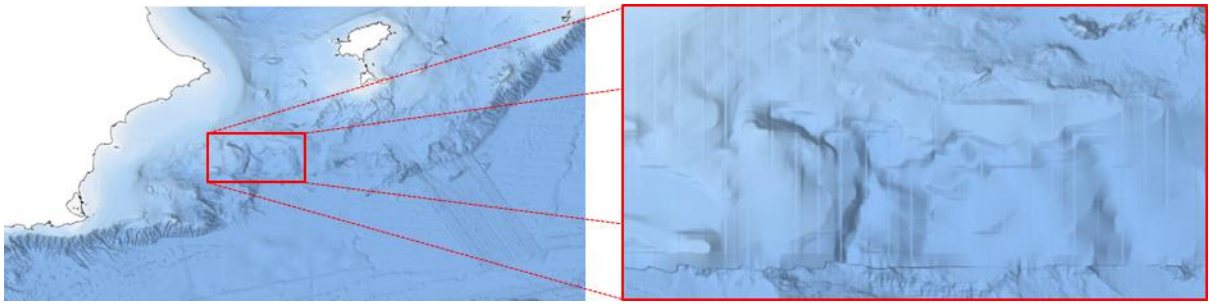

Figure 10. *Sample of a gridding artifact in the global DTM.*

This problem is well known in the state of the art as the *stripping* artifact. We started by reviewing the solutions in the literature [Crippen1989] [Oimoen2000] [Albani2003] [Tsai2008] [Dou2018] [Sun2019]. All the methods reviewed rely on first detecting the pattern and then correcting it. Note that, once the pattern is detected, any of the interpolation methods described in the previous sections can be used to correct it.

However, most of the methods assume the pattern is present in all the map, so detection is just reduced to finding visible lines. Also, most of them assume that this pattern is axis-aligned (i.e., the lines in the pattern are either vertical or horizontal). Both assumptions cannot be made in our case, since the composite DTM is constructed from different sources and the pattern just appears in a few areas and, due to further reprojections and rescaling of the original data during composition, the orientation of the lines is arbitrary.

The original idea was to develop a deep learning technique able to automatically identify this pattern. However, deep learning mechanisms require a large amount of training data (samples of how this pattern appears in different situations, for instance) and we do not have it. We identified some of the sources of data contributing to those errors. While these datasets are available at the portal, we received no response from the responsibles of the data, so we still do not have data to work with. A possibility to explore in the future is to generate this data synthetically. Obviously, this is not ideal, as this may leave out non-obvious cases that will not be detected by the system if we just use simulated data for training.

5. Feasibility of Implementing Fledermaus-like Navigation and Terrain Exaggeration in Cesium

Finally, Coronis also studied the feasibility of visualizing high resolution DTMs (HRDTM) within the web portal. The current web viewer relies on Cesium JS library to render the terrain tiles on the web browser. Cesium allows manipulating the 3D view in ways similar to fledermaus. In some of the internal meetings, we also discussed the possibility of adding markers or different layers of imagery, and Cesium allows both things transparently, and should not be a problem to develop a viewer that pops-up when the user selects a given HRDTM.

However, the main disadvantage of this library is that it does not allow the real-time change of the terrain exaggeration, according to these sources:

- https://community.cesium.com/t/activate-terrain-exaggeration/6098/2
- https://github.com/CesiumGS/cesium/issues/4342

So, after reviewing the possible options, we studied the different possibilities available to implement this feature without changing the core library. We list them in the following sorted by level of required effort:

- **Low effort**: make the terrain exaggeration "non interactive". Terrain exaggeration is set when creating the viewer object in Cesium, so we can let the user change that value and automatically trigger a *refresh* of the web page.
- **Mid effort**: create a 3D model for each HRDTM. According to this source, adapting the vertical size should be a matter of changing the *tile set transform*: https://github.com/CesiumGS/cesium/issues/7562
  There are available tools that we could adapt with low effort to transform a terrain into a 3D mesh in that format, e.g.: https://github.com/CesiumGS/obj2gltf.git
  However, this does not use the tiles we developed in the previous EMODnet project, the result of this conversion would be a single-file 3D model.
  If the size of those HRDTM is small enough so as to be transferred directly to the user for visualization, then this is the best option. Otherwise, we are facing the same problem we faced with the global DTM, and we need tiling mechanisms.
- **Huge effort**: there is a new capability of Cesium called 3D tiles: https://cesium.com/blog/2015/08/10/introducing-3d-tiles/
  As in the previous case, these 3D tiles allow also the scaling of the vertical coordinates using the *tile set transform*. However, as it happened with the terrain tiles, Cesium JS is providing the specifications, but not the software for creating them. This was overcome by using the software that was built by CORONIS in the previous project. .

6. References

[Albani2003] M. Albani and B. Klinkenberg. 2003. A Spatial Filter for the Removal of Striping Artifacts in Digital Elevation Models. Photogrammetric Engineering and Remote Sensing, 69, 755-765.

[Almansa2002] A. Almansa, F. Cao, Y. Gousseau, and B. Rougé. 2002. Interpolation of Digital Elevation Models Using AMLE and Related Methods. IEEE Transactions on Geoscience and remote sensing, vol. 40, no. 2, February 2002.

[Bertalmio2000] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. 2000. Image inpainting. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00). ACM Press/Addison-Wesley Publishing Co., USA, 417–424.

[Crippen1989] R. E. Crippen. 1989. A simple spatial filtering routine for the cosmetic removal of scan-line noise from Landsat TM P-Tape imagery, Photogrammetric Engineering & Remote Sensing, 55(3): 327–331.

[Delaunay1934] B. Delaunay. 1934. Sur la sphere vide. A la mémoire de Georges Voronoi. Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk (Bulletin of Academy of Sciences of the USSR), 7, pp. 793-800.

[Dou2018] H.-X. Dou, T.-Z. Huang, L.-J. Deng, X.-L. Zhao, J. Huang. 2018. Directional ℓ_0 Sparse Modeling for Image Stripe Noise Removal. Remote Sensing, 10, 361.

[Fasshauer2007] G. E. Fasshauer. 2007. Meshfree Approximation Methods with MATLAB. World Scientific Publishing. ISBN 978-981-270-633-1.
[Larsson2017] E. Larsson, V. Shcherbakov, A. Heryudono. 2017. A least squares radial basis function partition of unity method for solving PDEs. SIAM Journal on Scientific Computing 39 (6), A2538-A2563.
[Levin1998] D. Levin. 1998. The approximation power of moving least squares. Mathematics of Computation, Volume 67, 1517-1531.
[Oimoen2000] M. J. Oimoen. 2000. An effective filter for removal of production artifacts in U.S. Geological Survey 7.5-minute digital elevation models, Proceedings of the Fourteenth International Conference on Applied Geologic Remote Sensing, 06–08 November, Las Vegas, Nevada (Veridian ERIM International, Ann Arbor, Michigan), pp. 311–319.
[Shepard1968] S. Donald. 1968. A two-dimensional interpolation function for irregularly-spaced data. Proceedings of the 1968 ACM National Conference. pp. 517–524.
[Samet1984] H. Samet. 1984. The quadtree and related hierarchical data structures. *ACM Computing Surveys*. ACM. 16 (2): 187–260.
[Smith1985] G. D. Smith. 1985. Numerical solution of partial differential equations: finite difference methods (3rd ed.). Oxford University Press.
[Smith1990] W. H. F. Smith, and P. Wessel. 1990. Gridding with continuous curvature splines in tension. Geophysics, 55, 293-305.
[Sibson1981], R. Sibson. 1981. A Brief Description of Natural Neighbor Interpolation. Interpolating Multivariate Data. John Wiley & Sons: Nueva York, 21–36.
[Sun2019] Y.-J. Sun, T.-Z. Huang, T.-H. Ma, Y Chen. 2019. Remote Sensing Image Stripe Detecting and Destriping Using the Joint Sparsity Constraint with Iterative Support Detection. Remote Sensing. 11, 608.
[Tsai2008] F. Tsai and W. W. Chen. 2008. Striping Noise Detection and Correction of Remote Sensing Images. IEEE Transactions on Geoscience and Remote Sensing, vol. 46, no. 12, pp. 4122-4131.
[Wendland2002] H. Wendland. 2002. Fast evaluation of radial basis functions: Methods based on partition of unity. In: C.K. Chui et al. (Eds.), Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, 473–483.
[Wessel2009] P. Wessel. 2009. A general-purpose Green's function-based interpolator. *Comput. Geosci.* 35, 6, 1247-1254.
[Williams1998] C. K. I. Williams. 1998. Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond. Learning in Graphical Models. pp. 599-621.