Resampling of bathymetric data for SAS processing

Blair Bonnett[1], Holger Schmaljohann[2], and Thomas Fickenscher[1]

[1] Chair for High Frequency Engineering, Helmut Schmidt University, 22043 Hamburg, Germany
[2] Bundeswehr Technical Center for Ships and Naval Weapons, Maritime Technology and Research (WTD 71), 24340 Eckenförde, Germany

Corresponding author: *Blair Bonnett, Lehrstuhl für Hochfrequenztechnik, Fakultät für Elektrotechnik, Helmut-Schmidt-Universität, 22039 Hamburg, Germany; blair.bonnett@hsu-hh.de*

Abstract: *Generating synthetic aperture sonar (SAS) images involves delaying and coherently summing the echoes recorded by the elements of the receiving array over multiple pings onto the desired rendering points. This requires accurate knowledge of the two-way travel time taken by the signal to travel from the transmitter to a particular point and the echo to return to each physical receiving element at each ping within the synthetic aperture. The bathymetry of the scene has a significant impact on the two-way travel time which cannot be ignored. Inaccuracies in the assumed bathymetry, e.g., an incorrect height, lead to errors in the estimated travel time and cause defocusing of the image. If prior information of the bathymetry is available, it can be used to improve the estimate of the travel time. Typically it is desired to form the image on a grid with a regular sample spacing. However, the bathymetric data may not be regularly sampled. For example, our processing chain generates a pair of sidescan images using broadside beamforming with vertically separated receiver arrays to estimate the bathymetry via interferometry. Due to the non-ideal motion of the vehicle, the broadside beams are neither parallel to each other nor equally spaced. This effect is particularly pronounced when used with a circular geometry: bathymetric estimates will be dense in the centre of the circle and sparse near its circumference. There may also be gaps in the estimates due to shadows, area of low coherence, crabbing etc. Data captured externally (e.g., from an echo sounder, or on a prior mission) may be in a different coordinate system to the current mission. In this paper, we compare two methods for resampling the bathymetric measurements to an imaging grid, namely linear barycentric interpolation and inverse distance weighted (IDW) interpolation.*

Keywords: *synthetic aperture sonar, bathymetry, spatial resampling, linear barycentric interpolation, inverse distance weighted interpolation*

1. INTRODUCTION

Reconstructing synthetic aperture sonar (SAS) images requires the coherent combination of the echoes recorded on different pings, and thus an accurate knowledge of the two-way travel time of each acoustic pulse. If the bathymetry of the scene is available, it can be used to find the position of each focus point in the image to aid in calculating the travel time. This bathymetry could take the form of a digital elevation model (DEM, only the seafloor depths) or digital surface model (DSM, depths of the seafloor and any proud objects). It could be generated on a previous mission over the same area, however localising the trajectory of the system on the new mission within this data to a sufficient accuracy to generate high-quality images is difficult.

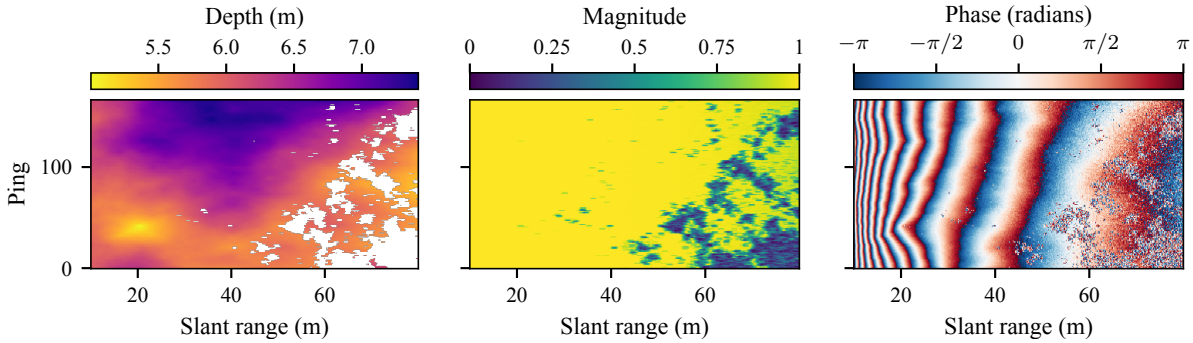
Alternatively, a DSM can be generated at an earlier stage of processing. In this paper we describe a sidescan processing scheme which use interferometry to estimate the bathymetry. This data is likely to have a lower along-track resolution than the final SAS image and, since it is measured broadside to the system, will not be sampled on a regular spatial grid due to non-ideal motion of the system. Areas of low coherence (e.g., shadows) will result in missing data points. As a result, using these datasets to provide bathymetric information for SAS reconstruction requires resampling to the desired imaging grid and interpolation to fill in missing areas.

In this paper, we compare two methods for resampling bathymetric data: linear barycentric interpolation and inverse distance weighted (IDW) interpolation. Section 2 describes the sidescan interferometry process which generates the real input data and a method for generating test data with ground truth. The interpolation methods are then presented in Section 3. They are compared in terms of interpolation accuracy and computation time in Sections 4 and 5, respectively. We conclude with a discussion in Section 6.

2. SIDESCAN INTERFEROMETRY

Our processing toolchain includes an interferometric sidescan step to estimate the bathymetry of the imaged scene. An initial ground plane is provided by the operator and a simple delay-and-sum beamformer is used to focus the energy onto this plane at points intersecting a beam projected broadside from the vehicle. This results in a stripmap-style sidescan image indexed by ping number and slant range. The complex coherence between the images from both arrays of an interferometric pair (generated using the same focal points) is calculated within 1D windows in slant range over a number of pixel shifts. The fractional shifts leading to the highest coherence magnitude at each point are estimated using peak fitting and form an interferogram giving the phase shift observed between the two arrays. This can then be unwrapped and the depth and across-track position of each original point subsequently updated to give a bathymetric dataset of (x, y, z) positions indexed by ping number and slant range [1].

Evaluating the performance of the remapping requires ground truth of the scene. As this is not available for real data, a synthetically generated bathymetric dataset is used. First, the stochastic subdivision ('diamond-square') algorithm introduced by Fournier et al. [2] is used to create a DSM. A two-dimensional array of depths is iteratively subdivided; after L iterations, a side containing M initial points is increased to $2^L(M-1)+1$ points. The depths of the new points at each iteration are taken from a normal distribution $\mathcal{N}(\mu, c^{-lH})$ where μ is the mean of the surrounding points, l is the iteration number, H is a self-similarity parameter and c is a constant. Each iteration adds features of increasingly higher spatial frequency with a decreasing magnitude, retaining the general shape set by the earlier iterations. This is impractical for large L



(a) Bathymetry *(b) Coherence map* *(c) Interferogram*

Figure 1: Bathymetric dataset generated using $L = 8$, $c = 2.5$, $H = 0.87$, a baseline of 0.2 m, a wavelength of 1 cm, a SNR decreasing from 40 dB to 10 dB with range, a minimum grazing angle of 5°, a window size of 51 points and a coherence threshold of 2/3.

due to the large number of new points. However, the generating distribution approaches $\mathcal{N}(\mu, 0)$ as l increases, meaning that later iterations are effectively equivalent to bilinear interpolation.

A bathymetric dataset can then be generated from the DSM. First, a 1D slice is obtained by interpolating along a line from the horizontal position of the system. A second interpolation then yields the x, y and z values as a function of slant range from the system. The differential phase between the receivers can be calculated for a given interferometric baseline and wavelength [3]. Phase noise based on the expected SNR is added, and ranges corresponding to shadowed areas or areas of low grazing angle have their phase set to a random value between $\pm\pi$. The local variance of this noisy interferogram is calculated with a sliding window in slant range. A lookup table is then used to generate a coherence map from the phase variance. Applying a threshold to this allows areas of the bathymetry where no valid data would be found by the sidescan processing to be masked out. Fig. 1 shows an example of a generated bathymetric dataset.

3. INTERPOLATION METHODS

As the measured bathymetry will not be sampled on a regular grid, simple methods such as bilinear interpolation cannot be used. Two methods are compared here: linear barycentric interpolation and inverse distance weighted (IDW) interpolation. Initially, a radial basis function (RBF) interpolator was also tested. Similar to the IDW method, this used a weighted sum of RBFs from the surrounding points to calculate the output depth. However, this was significantly more computationally expensive while not improving the accuracy of the interpolated bathymetry, and so is not included in this comparison.

The linear barycentric interpolator first computes the Delaunay triangulation of the input points. The barycentric coordinates of a target point in the triangle containing it are used to generate weights from each corner of the triangle. These are used in a weighted sum of the measured depths at the corners to calculate the interpolated depth. The SciPy library [4] contains an implementation of this method which was used for this comparison.

Introduced by Shepard [5] for the interpolation of spatial data, the IDW method computes the output as the weighted sum of the inputs. The basic weighting function given by Shepard is

$$w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^p}, \tag{1}$$

where the subscript i indexes the input data samples, $\mathbf{x}$ is the output coordinate, $\mathbf{x}_i$ is the i^{th} input coordinate, d is a distance metric between two coordinates and p is the weighting exponent. The Euclidean distance and $p = 2$ are used throughout this paper. For computational efficiency, our implementation uses SciPy's k-d tree to select the N closest points instead of weighting all points.

4. INTERPOLATION ACCURACY

To test the accuracy of the presented interpolation methods, three scenarios were modelled: a linear trajectory past an empty seafloor with a low height variation causing no shadows, a linear trajectory past proud objects casting shadows, and a circular trajectory around a scene with a proud object. A sonar with a vertical interferometric baseline of 0.2 m, an operating frequency of 150 kHz and a ping-to-ping spacing of 30 cm was simulated. The sound speed was 1500 m/s, and the coherence was calculated over bins of 51 samples in length with a magnitude threshold of 2/3. All bathymetric datasets included a beam broadside to the vehicle and beams 5° either side of this. There were no depth errors added to the generated bathymetry; in a real-world case, the interferometry would not be perfectly accurate, but for this study we are only interested in the performance of the interpolators. The IDW interpolations use the 24 nearest neighbours up to a maximum Euclidean distance of 2 m from the target point.

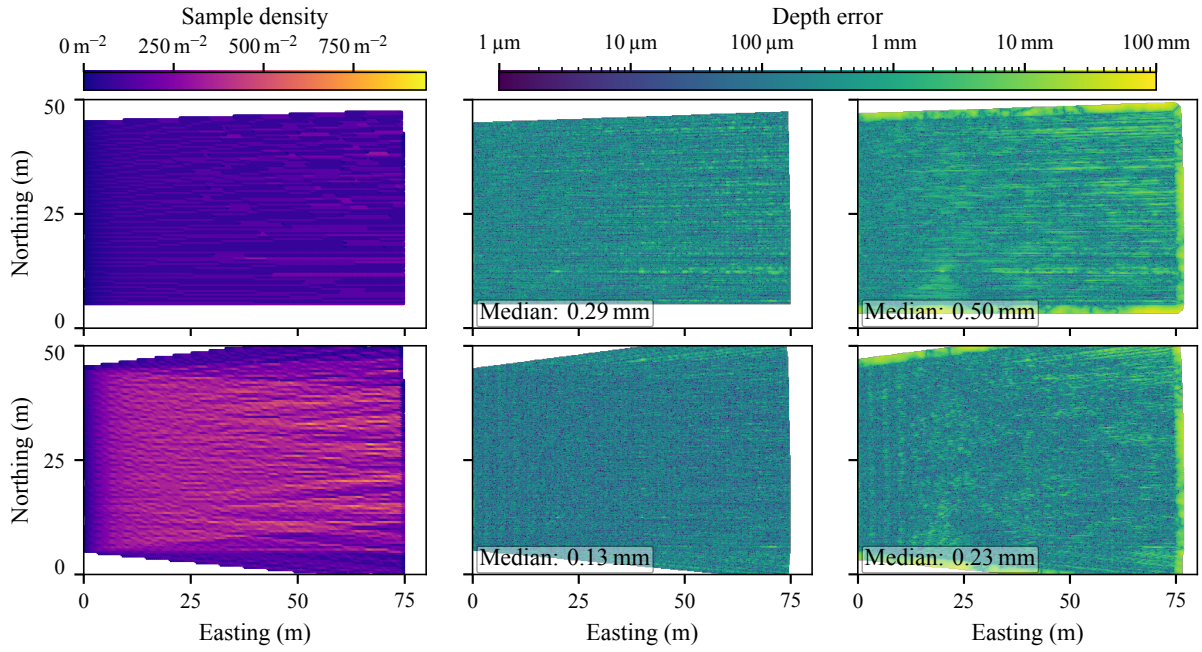
For the first test, a 50 m × 80 m DSM was generated along with a 40 m linear trajectory heading north along the western edge of the DSM. Sway was included in this trajectory by using a random walk for the system heading with a ping-to-ping heading variance of 0.05°. The DSM has depths ranging from 5.86 m to 6.49 m with shallow slopes so that there are no missing areas from shadowing. The sample density of the resulting bathymetric dataset is shown in Fig. 2(a) for both the broadside beam only and for all three beams. The depth errors after remapping both cases to the original DSM coordinates with the linear barycentric and IDW interpolators are shown in Figs. 2(b) and 2(c), respectively. Note that the IDW method extrapolates up to the maximum distance allowed for input points.

In all four cases, the errors in the regions fully covered by the beams broadly follow a negative exponential distribution with the parameter $\lambda = \ln 2 / \operatorname{median}(e)$. The error from both interpolators is unsurprisingly correlated with the density of the input samples: areas where there are more points lead to lower errors. The median errors back this up: using three beams for bathymetry results in a lower error than a single beam due to the increased sample density. The linear barycentric interpolator has lower errors in general than the IDW interpolator.

Another DSM was then created with two proud objects added: a 1 m radius cylinder and a 1 m × 2 m rectangular pyramid, both reaching 1 m above the seafloor (Fig. 3(a)). Again, a swaying trajectory with a ping-to-ping heading variance of 0.05° was used in generating the bathymetry. This was then remapped to the original DSM coordinates using both linear barycentric and IDW interpolators with the resulting errors shown in Figs. 3(b) and 3(c), respectively.

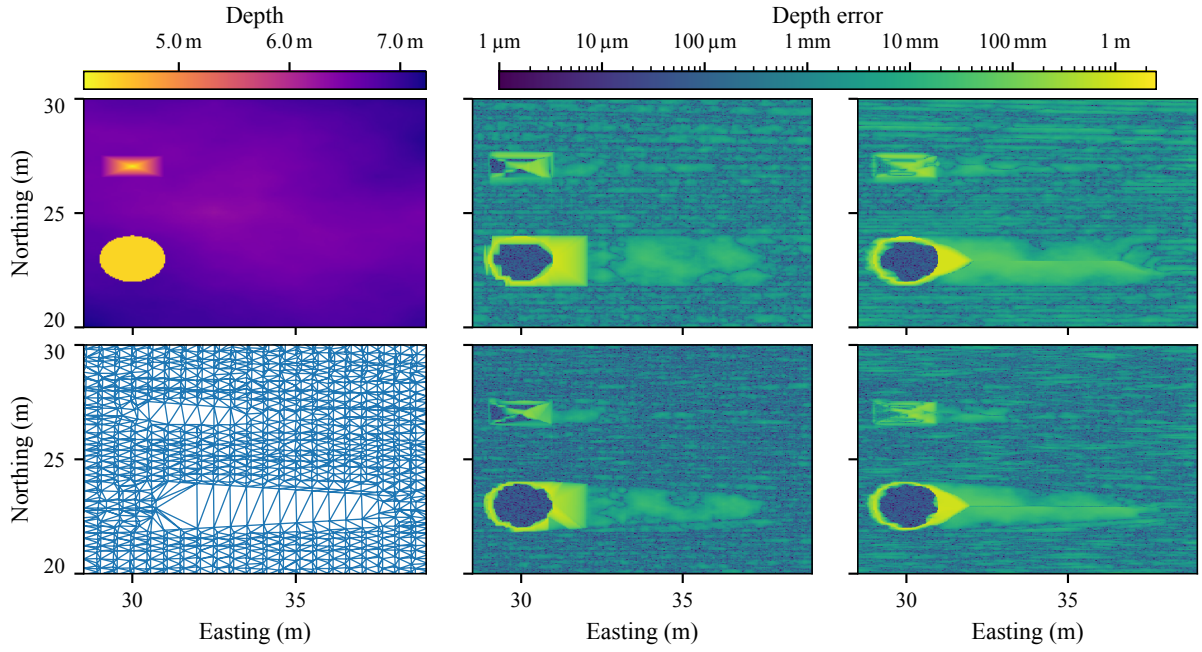
As in the previous example, using three beams results in lower errors than a single beam. The linear barycentric interpolator again has a better performance than the IDW interpolator. As well as the surrounding seafloor, this is especially noticeable on the faces of the pyramid (apart from the occluded eastern face). The vertical sides of the cylinder cause problems for both interpolators. In a real-world scenario, layover around the vertical faces would cause errors in the bathymetry in these locations anyway.

The shadow behind the cylinder leads to a rectangular patch of high error with the linear barycentric interpolator, while the IDW interpolator has a more expected cone shape. This can



(a) Bathymetry density *(b) Linear barycentric error* *(c) IDW error*

Figure 2: Resampling a plain seafloor with a broadside beam (top row) and three beams (bottom row). (a) Density of input bathymetric samples measured over 0.5 m × 0.5 m cells. (b) and (c) Error between interpolated outputs and the original DSM. The annotated medians are calculated over the fully covered regions.



(a) DSM and triangulation *(b) Linear barycentric error* *(c) IDW error*

Figure 3: Resampling a scene with a 1 m radius cylinder and a 1 m × 2 m pyramid, both 1 m tall. (a) The original DSM and a simplified triangulation of the three beam bathymetry. (b) and (c) The difference between the original DSM and the linear barycentric and IDW interpolators, respectively (top: broadside only, bottom: three beams).

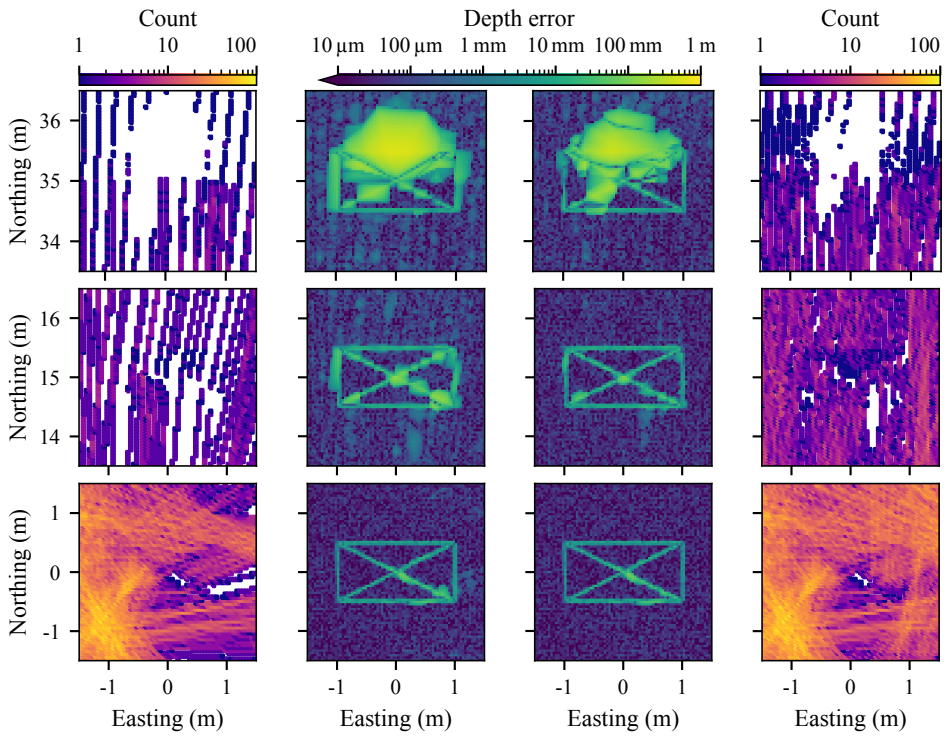
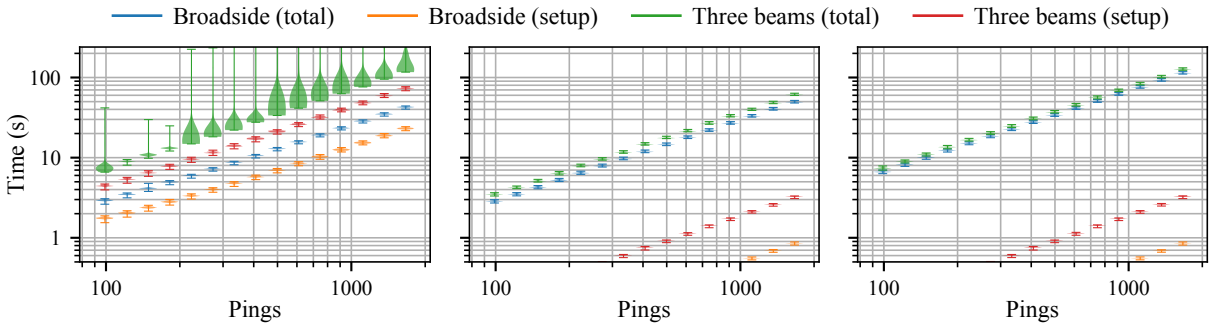


Figure 4: Resampling using a linear barycentric interpolator on bathymetry collected on a 40 m radius circular trajectory. A 1 m × 2 m pyramid is at the circle centre (bottom), 15 m north of the centre (middle) and 35 m north of the centre (top). The left two columns show the number of input points in 5 cm × 5 cm blocks and the error from the original DSM for a single beam. The right two columns repeat this for three beams.

be explained by the triangulation of the bathymetric data presented in Fig. 3(a) (for display purposes the bathymetry has been downsampled by a factor of 20 in slant range – the full triangulation exhibits the same patterns). The triangles joining the samples from the first beams passing the cylinder have a vertex on the edge of the cylinder, biasing the interpolation.

Finally, a circular trajectory with a 40 m radius was created. Similarly to the previous examples, the ping-to-ping heading change had an error with variance 0.05° added to the change required to follow the circle. A 1 m × 2 m rectangular pyramid standing proud of the seafloor by 1 m was placed at various distances from the centre of the trajectory. Fig. 4 shows the input sample density and the error between the linear barycentric interpolated output and the original DSM around the pyramid for three positions, comparing the single- and three-beam cases. Although not presented in the figure, the IDW method was tested with this dataset. As seen in the linear case, it was not able to accurately output the sloped faces of the pyramid.

Again, the accuracy of the interpolation is dependent on the density of the input points. Note that with a 10° beamwidth and a 40 m radius trajectory, the full-view area of the reconstructed images would be a circle with a radius of 3.5 m. The interpolation is accurate within areas outside this full-view area as it does not rely on forming a synthetic aperture from all angles, but rather just needs a few beams to cross the area of interest. In many situations, a single bathymetric beam crossing the full-view area is likely to be sufficiently accurate. Taking into account the vertical opening angle may limit the minimum and maximum ranges the bathymetry can be calculated for with a real system, and thus reduce the area that can be accurately interpolated. It is anticipated that most systems will have an opening angle larger than that required to cover the full-view area, allowing the interpolation to be performed outside the full-view area.



(a) Linear barycentric *(b) IDW ($N = 8$)* *(c) IDW ($N = 24$)*

Figure 5: Violin plots of the total interpolation time and the setup (triangulation or k-d tree creation) component for different interpolators when upsampling the input data by a factor of 12 in the along-track direction.

5. INTERPOLATION TIME

To compare the time required to interpolate the bathymetric data with the two methods, a number of timing trials was performed. For each trial, a unique DSM was created and an ideal linear trajectory with a ping-to-ping spacing of 30 cm was used to generate bathymetric data over a slant range of 7 m to 70 m with a range resolution of 2.5 cm. A subset of the first P pings was taken and interpolated onto a regularly spaced grid of the same along-track length as the input subset and from 0 m to 70 m across-track at a resolution of 2.5 cm × 2.5 cm, i.e., an along-track upsampling by a factor of 12. It is expected that most real-world systems would have a similar upsampling factor. This was performed for 15 different values of P ranging from 100 to 1667 pings (corresponding to trajectory lengths of 30 m to 500 m) for 400 trials.

Violin plots of the time taken for interpolation is shown in Fig. 5. For the barycentric interpolation, the time taken to triangulate the input takes the majority of the total time. By contrast, for the IDW interpolator the k-d tree construction time is minimal and the bulk of the time is taken by selecting and summing the neighbours. The linear barycentric method is approximately equal to the IDW method with $N = 24$; IDW with $N = 8$ is about 40 % faster.

There are some outliers for the three-beam barycentric interpolator where the total time is significantly larger than average. Investigation of these cases showed that these were a result of trying to interpolate points close to but outside the convex hull of the input dataset. In this case, the interpolator was slow when searching for a triangle to interpolate on. Points further away from the convex hull did not exhibit this slowdown. It is uncertain whether this is a general problem with the method or with the implementation used here, but the latter is suspected.

6. DISCUSSION AND CONCLUSION

For the IDW method, the number of input points N is an important parameter. The ping-to-ping spacing of the modelled system was twelve times greater than the range resolution, a factor that we consider realistic for a typical SAS. As a result, the twelve bathymetric measurements nearest to a target point are likely to be from the same beam, and so for $N \leq 12$ the interpolation would tend to be one-dimensional along the nearest beam. Our choice of $N = 24$ for most of the testing ensures that multiple beams are used. A lower value of N may reduce the amount

of smoothing and better capture local depth variations. The value of N should be tuned for the collection geometry of a given system, especially if multiple beams are used.

The linear barycentric method gave a more accurate interpolation in all scenarios. For simple scenes, the errors from the IDW method are likely to be negligible compared to errors in the bathymetry estimates which are not modelled in the test datasets. The linear barycentric method was clearly superior when objects were included in the scene, which is not surprising as the triangulation is able to capture the geometry of the objects better than the weighted average that is used for the IDW method. Shepard [5] gives an alternative weighting function taking the slope into account which could help at the cost of increased computation time. Although neither method could interpolate the shadowed areas behind proud objects, this is of no concern as there will be no valid data to reconstruct an image in these areas. For $N = 24$ the two methods are essential equivalent in terms of computation time, while the IDW method is faster for lower N, although likely by not enough (especially when compared to the total time taken by the full SAS processing chain) to offset the reduced accuracy.

IDW interpolation will have a higher memory usage than linear barycentric. The latter must store the vertex indices of each triangle, and during interpolation calculate three barycentric coordinates and the corresponding weights for each point. The IDW interpolator must find N points and their distances from the target point, and then calculate the weights. As typically $N > 3$, this leads to a higher memory usage. In practice, the memory requirements may require the interpolation to be performed in separate chunks, especially for long missions.

Parallelisation could reduce the time required to perform the interpolation. For the linear barycentric method, the triangulation of the input dataset is the bottleneck. Some methods have been published to allow this triangulation to be performed in parallel blocks while maintaining the consistency of the triangulation across the boundaries of the blocks [6]. The interpolation step could also be performed in parallel after the triangulation is complete. By contrast, the construction of the k-d tree is significantly faster than the subsequent IDW weighted sum. The SciPy implementation used here allows multi-threaded selection of points from the tree which speeds up the timing tests by $\sim 400\,\%$ when using 8 threads. The construction of the k-d tree could also be parallelised by assigning each branch to a different thread after splitting.

REFERENCES

[1] T. O. Sæbø. “Seafloor Depth Estimation by means of Interferometric Synthetic Aperture Sonar”. PhD thesis. University of Tromso, 2010.

[2] A. Fournier, D. Fussell, and L. Carpenter. “Computer rendering of stochastic models”. *Commun. ACM*, vol. 25, no. 63, June 1982, pp. 371–384.

[3] M. Eineder. “Efficient simulation of SAR interferograms of large areas and of rugged terrain”. *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 6, 2003, pp. 1415–1427.

[4] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nat. Methods*, vol. 17, Mar. 2020, pp. 261–272.

[5] D. Shepard. “A two-dimensional interpolation function for irregularly-spaced data”. *Proceedings of the 1968 23rd ACM National Conference*. ACM, 1968, pp. 517–524.

[6] C. Nguyen and P. J. Rhodes. “TIPP: Parallel Delaunay Triangulation for Large-Scale Datasets”. *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. ACM, 2018.