FURTHERING USABILITY AND EFFICIENCY OF MASSIVELY PARALLEL GAUSSIAN PROCESS REGRESSION AS APPLIED TO MULTI-BEAM SONAR DATA

BY

PHILLIP M. PARISI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

OCEAN ENGINEERING

UNIVERSITY OF RHODE ISLAND

2023

MASTER OF SCIENCE THESIS

OF

PHILLIP M. PARISI

APPROVED:

Thesis Committee:

Major Professor Chris Roman

Mingxi Zhou

Peter Swaszek

Brenton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2023

ABSTRACT

To be completed at a later date when it is important.

ACKNOWLEDGMENTS

This thesis would not be possible without unique insights and guidance from my advisor, Chris Roman, and committee members Mingxi Zhou and Peter Swaszek. This thesis furthers Kristopher Krasnosky's dissertation; Kris' time and effort were crucial to success. Dr. Tzu-Ting Chen provided a valuable perspective on sonar uncertainty, and I am grateful for her advice. Last but not least, thank you to my fellow lab mates for keeping me focused and laughing along the way.

PREFACE

This thesis is presented in manuscript form.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iii

PREFACE iv

TABLE OF CONTENTS v

MANUSCRIPT

1 Furthering Usability and Efficiency of Massively Parallel Gaussian Process Regression as Applied to Multi-Beam Sonar Data 1

1.1 Introduction 2

1.1.1 Gaussian Process Regression 2

1.1.2 Massively Parallel Gaussian Process Regression 3

1.1.3 Approximate Methods 5

1.2 Implementation 6

1.2.1 System Overview 8

1.2.2 Downsampling 9

1.2.3 Hyperparameter Training 13

1.2.4 Incorporating Uncertainty 14

1.3 Results 15

1.4 Discussion 18

1.4.1 Future Work 19

1.5 Conclusion 20

Page

LIST OF REFERENCES 22

APPENDIX

A Notation and Relevant Equations 24

B Significant Parameters 27

C Downsampled Swaths 30

MANUSCRIPT 1

Furthering Usability and Efficiency of Massively Parallel Gaussian Process Regression as Applied to Multi-Beam Sonar Data

by

Phillip M. Parisi[1], Chris Roman[1,2], Kristopher Krasnosky[3]

In preparation for submission to the Journal of UNKNOWN

[1] University of Rhode Island, Department of Ocean Engineering
[2] University of Rhode Island, Graduate School of Oceanography
[3] University of South Florida, Center for Ocean Mapping and Innovative Technologies

1.1 Introduction

Multi-beam echo sounder (MBES) surveys repeatedly ping the seafloor using acoustic transducers to determine the water depth at points along the seafloor. These returns (pings) can contain upwards of one hundred data points, and surveys that ping the seafloor for hours will result in millions of data points. Forming models of the seafloor from these data can be viewed as a regression problem. Regression, determining the relationship between a dependent variable and one or more independent variables, can be solved using Gaussian Process Regression (GPR) [1]. GPRs can be used to generate water depth predictions over a dense grid of points along the seafloor and calculate the associated uncertainties to provide a level of confidence in the predictions. Modern robotic algorithms that utilize probabilistic approaches benefit from GPR's uncertainty estimates in decision-making and estimation tasks. GPR has been applied to underwater terrain modeling [2], terrain aided navigation (TAN) [3], and simultaneous localization and mapping (SLAM) [4, 5]. It is inefficient to compute GPR on large datasets; thus, utilizing a graphical processing unit (GPU) to reduce compute time has been explored [6, 7]. Krasnosky's Massively Parallel Gaussian Process Regression (MP-GPR) agorithm was shown to improve the resolution of seafloor terrain models made from MBES data and generated an uncertainty model which other standard mapping methods (splines, gridded, polynomial, etc.) lack [5]. This work progresses Krasnosky's research by considering additional computational efficiencies and uncertainty from multi-beam sonar.

1.1.1 Gaussian Process Regression

GPR is a supervised machine learning approach for determining input-output mappings from training data. The basic formulation of the problem uses empirical data points D from a set of N observations, known as *training data*, consisting

of inputs $\boldsymbol{x}$ (independent variables) and the corresponding targets $\boldsymbol{y}$ (dependent variable). Each observation $\boldsymbol{x}_i$ of d dimensions maps to a scalar target y_i, which is a noisy realization of the underlying function of interest f. That is,

$$D = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, N\}$$
$$\boldsymbol{x} \in \mathbb{R}^{N \times d}, \boldsymbol{y} \in \mathbb{R}^N, y_i = f(\boldsymbol{x}_i) + \epsilon,$$

where ϵ denotes error and bold symbols indicate matrix quantities. The error comes from sensor observation and is often assumed be normally distributed with zero mean, such that $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. When applied to seafloor mapping, $\boldsymbol{x}_i = (\text{northing}, \text{easting})$ are the position variables, $y_i = \text{depth}$ is the corresponding depth at that location, and f is the 'function' that describes the 2.5-dimensional seafloor surface. During the training portion of the problem, D is used to learn the underlying function f. During the inference step, f is used to predict depths $\boldsymbol{y}_* \in \mathbb{R}^{A \times 1}$ over a new set of geographically dense inputs $\boldsymbol{x}_* \in \mathbb{R}^{A \times d}$ to form a point cloud map. A given prediction value, y_*, is fully described by a Gaussian distribution $y_* \sim \mathcal{N}(\mu_*, \sigma_*^2)$ with a mean depth value μ_* and variance σ_* (uncertainty).

GPR is a non-parametric regression model that utilizes a kernel to describe the relevant length scale and process noise of the function being modeled. A commonly used kernel is the squared exponential (SE) kernel,

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 exp(-\frac{|\boldsymbol{x} - \boldsymbol{x'}|^2}{2l^2}), \tag{1}$$

where $|\boldsymbol{x} - \boldsymbol{x'}|$ denotes the magnitude of the difference between vectors $\boldsymbol{x}$ and $\boldsymbol{x'}$ (Euclidean distance), and $\boldsymbol{\theta} = \{l, \sigma_f^2\}$ are hyperparameters determined during model training. The set of GPR equations are provided in the Appendix.

1.1.2 Massively Parallel Gaussian Process Regression

Krasnosky's Massively Parallel GPR (MP-GPR) is an extension of GPR that creates an updateable 2.5D model of the seafloor using multi-beam sonar data

collected in a push-broom manner. The outputted models are dense point clouds with uncertainties that can be used as terrain models for SLAM and other applications. Most modern regression methods lack uncertainty estimates and fail to inform users of the confidence bounds in prediction points, whereas MP-GPR does.

The underlying computation of a GPR suffers from mathematical inefficiencies when training on large datasets, primarily due to the required inverse of a $N \times N$ covariance matrix which faces cubic time complexity, $O(N^3)$. This renders the basic implementation of GPR intractable for massive data volumes [1]. Multibeam sonar surveys often generate millions of data points and hence cannot be computed using standard GPR.

MP-GPR relies on a parallel computational framework to speed up the computation. Its updateable GPR model leverages parallel processor cores on a GPU, iterative matrix updates that incorporate new bathymetric data when collected, sparse matrix representation, and an exactly sparse approximation to the SE kernel,

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \begin{cases} \sigma_f^2[\frac{2+cos(2\pi\frac{d}{l})}{3}(1-\frac{d}{l}) + \frac{1}{2\pi}sin(2\pi\frac{d}{l})] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases} \tag{2}$$

as devised by Melkumyan and Ramos [8]. This modified kernel closely approximates the actual squared exponential kernel with the added benefit of setting covariances to zero beyond the trained length scale. More zero elements in covariance matrices improve the performance of sparse matrix representations that MP-GPR employs.

These modifications allow for a tractable, online, and exact GPR solution that generates an in-situ terrain model with uncertainties. MP-GPR can operate

in real-time but requires significant computing power (e.g. a NVIDIA 2080ti GPU with 4352 CUDA cores). Modern mobile platforms, such as autonomous underwater/surface vehicles (AUVs/ASVs), are unlikely to possess high-end GPUs due to their significant power consumption. With ever increasing sensor resolution and acquisition rates, there is a need to further reduce the GPR compute time to enable real-time performance on a wide-range of hardware, such as the smaller NVIDIA Jetson NX Xavier, to make MP-GPR broadly applicable and relatively platform agnostic.

1.1.3 Approximate Methods

GPR and MP-GPR calculate an *exact solution*, that is, a solution that incorporates every data point in D for model training and inference. Alternatively, approximate methods have been developed to reduce compute time and calculate an *approximate solution* to GPR. There are many families of approximate methods: subset of regressors [9], subset of data [10], projected processes [11], and modern methods such as Sparse Variational Gaussian Processes [12].

Most approximate methods present alternative formulations to the underlying GPR equations and diverge from MP-GPR's implementation. However, data-centric methods incorporate techniques to replace training data D with a fabricated dataset D_R containing M points, where $M \ll N$, to reduce the computational burden. D_R can be composed of a subset from the training data D or of newly generated pseudo-data points that may not be present in D, such that $D_R \not\subset D$. D_R then serves as the training data for the model in lieu of D, decreasing the algorithm time complexity from $O(N^3)$ to $O(M^3)$ and reducing model accuracy to various degrees. Determining D_R from D will be broadly referred to as *downsampling*. Combining downsampling approaches proposed in approximate GPR algorithms and data science with MP-GPR results in a faster online GPR solution.

1.2 Implementation

MP-GPR was implemented using the Robotic Operating System (ROS) architecture [13], C++ code, and CUDA [14] code for GPU operations. The work described in this thesis added to the original code base and was tested on a pre-recorded ROSbag to simulate real-time operation.

Each downsampling method was tested by running a simulated live bathymetry survey. The process is:

- a sonar ping is obtained
- outliers are removed from the ping
- the ping is downsampling per a given method and downsample percent
- uncertainty values are calculated for each data point in the ping
- multiple pings are queued into a block for processing
- once a block is full, perform GPR inference to create a tile (dense set of prediction points)
- align tiles to make a map

After a run is performed, a point cloud model of the seafloor remains. The time to run each method was recorded, and the accuracy of each model can be computed through comparison with the model from MP-GPR using every datapoint (no downsampling).

The dataset used in this study was collected with a WASSP sonar system in the St. Mary's River (Georgia, USA). The WASSP Multibeam Sonar retrieves a maximum of 256 points per ping at 5-20Hz depending on depth. The river was well mixed and the measured sound speed profile was nearly constant with depth, alleviating the need for ray tracing corrections. The mapping system and survey details are fully explained by Krasnosky, et. al [15].

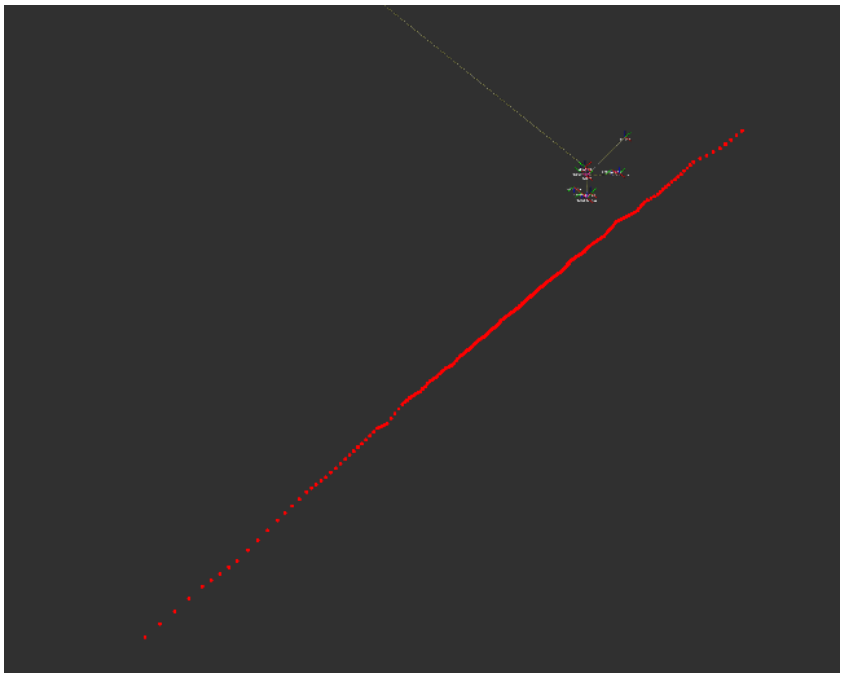

Figure 1. Hundreds on individual sonar soundings (red dots) form a ping.

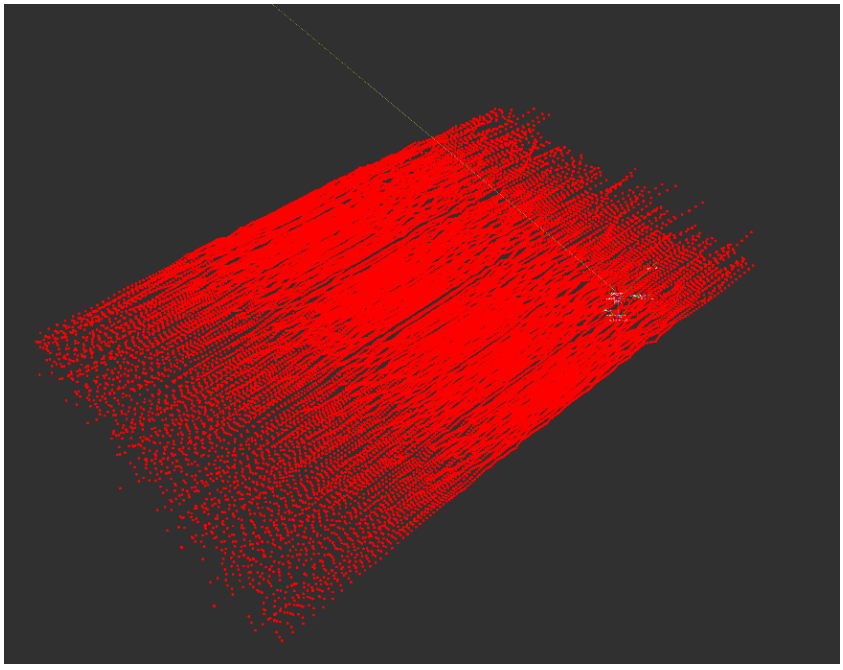

Figure 2. Many consecutive sonar pings form a swath.

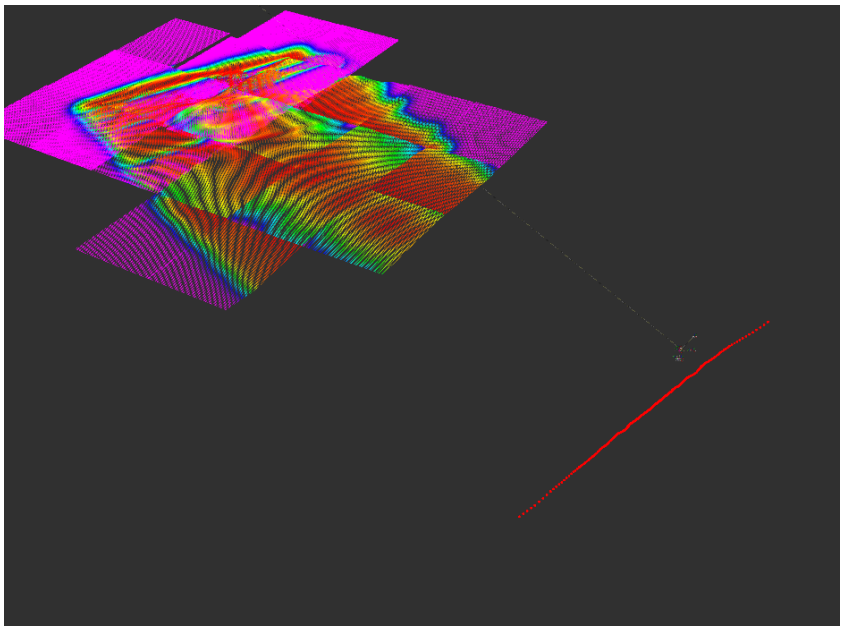

Figure 3. A group of GPR inference tiles with local heatmaps are calculated and aligned.

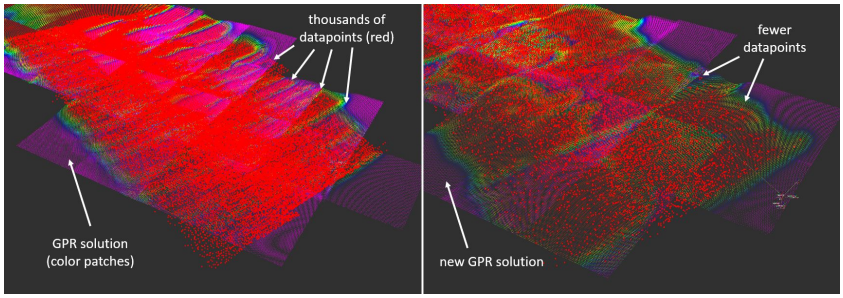

Figure 4. Snapshot of ROS Rviz simulation running MP-GPR on a raw sonar swath (left) and with a 10% random downsampled swath (right). Individual soundings (data points) are shown in red. GPR solution tiles are shown in color.

1.2.1 System Overview

The system ingests a live sonar data-stream one ping at a time. A navigation solution paired to each sonar ping accounts for dynamic effects and positions the data in the correct mapping frame. Sonar pings are passed to a 'blockulator.' The blockulator applies an outlier filter to remove erroneous soundings, downsamples the data, and queues it into a 'block.' Once a local neighborhood of blocks are

full, GPR inference is performed on to generate a large prediction tile. These tiles are added to the seafloor model as they become available during the survey [5].

Each downsampling method was implemented within a blockulator. The downsampling methods analyzed were:

- a *Uniform Random* method that selects points using a uniform random distribution,
- a *Systematic Random* method that selects every jth point (decimation),
- a *Random Hybrid* method that evenly combines systematic and uniform random selection,
- a *Point Averaging* method that averages dense local points into a set of sparse points,
- a *Dissimilar Neighbor* method that ranks points based on a ratio of nearest-neighbor distances to select 'interesting' points,
- and a *K-means* method that clusters points into groups based on similarity.

1.2.2 Downsampling

The downsampling methods are cheap to compute and implemented on raw bathymetry data as it streams in during a multi-beam survey. Each ping of sonar data is downsampled when received (D with N points into D_R with M points), agnostic to other processes and data collected. Downsampled pings are queued into blocks which are processed periodically and added to the GPR solution. The resulting time complexity is approximately $O(M^3)$.

When downsampling, it is of utmost importance that no outliers are included in D_R. The WASSP has a built-in filter to remove significantly irregular points, yet poor points may persist. Thus, an additional outlier filter was added before downsampling. A given point p was considered an outlier if its depth value exceeded 2.0x the standard deviation of the neighboring points' mean (excluding p). This

is especially relevant for the dissimilar neighbor method, which is biased towards outlier inclusion. Thus, outliers were removed from the set, and the clean D_R was passed to a downsampling algorithm.

Uniform Random

Uniform random sampling is a common and fast method for downsampling. For each sonar ping, a new random seed was used. Uniform random sampling is often a benchmark for evaluating the performance of other methods.

Systematic Random

Systematic random sampling ('decimation') sorts the data by the spatial y-coordinate (left to right along a ping) and selects every jth point along that dimension. In MATLAB, the indices are expressed $x_o : j : x_{end}$. To avoid along track selection bias, the first index of each ping x_0 was uniform randomly selected from 0 to $j - 1$.

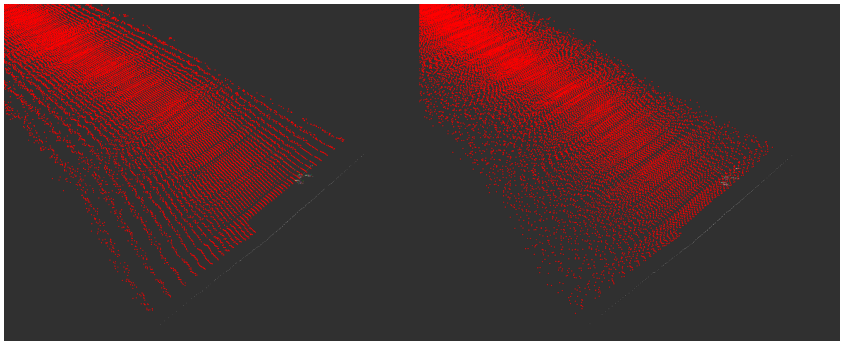

Figure 5. 20% systematic downsampling with biased (left) and unbiased (right) approaches

Hybrid Random

The hybrid random method recognizes the combined value of systematic samples to ensure a spatial spread and random sampling to select unique points. Thus, half of D_R is selected spatially with systematic sampling. Then, the second half is

selected from the remaining points using a uniform random distribution.

Point Averaging

Point averaging takes the average of local data points to create a new representative data point. This is done using a moving window across each ping.

Dissimilar Neighbor

The k-nearest-neighbor (KNN) algorithm is commonly used in machine learning for classification and regression. KNN uses a difference metric to determine if a new point is more similar to one group than another. The base assumption is that points with small nearest neighbor distances are more likely to be similar to those neighbors [16].

In the spirit of seeking 'unique' points to represent the distribution of a sonar ping, dissimilar neighbor (DN) checks the two neighboring points of p_i along a ping, p_{i-1} and p_{i+1}, and calculates Euclidean distances d_p. Points that have the smallest distance ratio $d_r = min(d_p)/max(d_p)$ become inclusion points. A point with a ratio close to 1 is evenly-spaced with its neighbors and assumed to be uninteresting, and vice versa. A ratio is used to scale distances as the spacing of points varies along a ping.

K-means

K-means is a popular clustering algorithm that splits a group of samples into k-clusters (subgroups) based on their similarity. The inclusion points chosen by uniform random sampling were used as the initial k-means centroids to encourage unique point distributions. To combat performance issues [17] and keep processing time minimal, the number of algorithm iterations was set to 15 [PHIL CHECK THIS]. Then, the centroids of each cluster served as the new data points in D_R.

Downsampling Implementation

Each method has one adjustable parameter, *downsample percent* s, which is the percent of points that *remain* after downsampling a given ping (e.g. if $s = 0.2$, a 200-point ping is downsampled to 40 points). A small value of s reduces the amount of data, decreases computation time, and may decrease solution accuracy. Conversely, a larger value of s includes more data, increases computation time, and may improve solution accuracy. s was systematically varied for each method to determine the optimal value that minimizes compute time while maximizing solution accuracy.

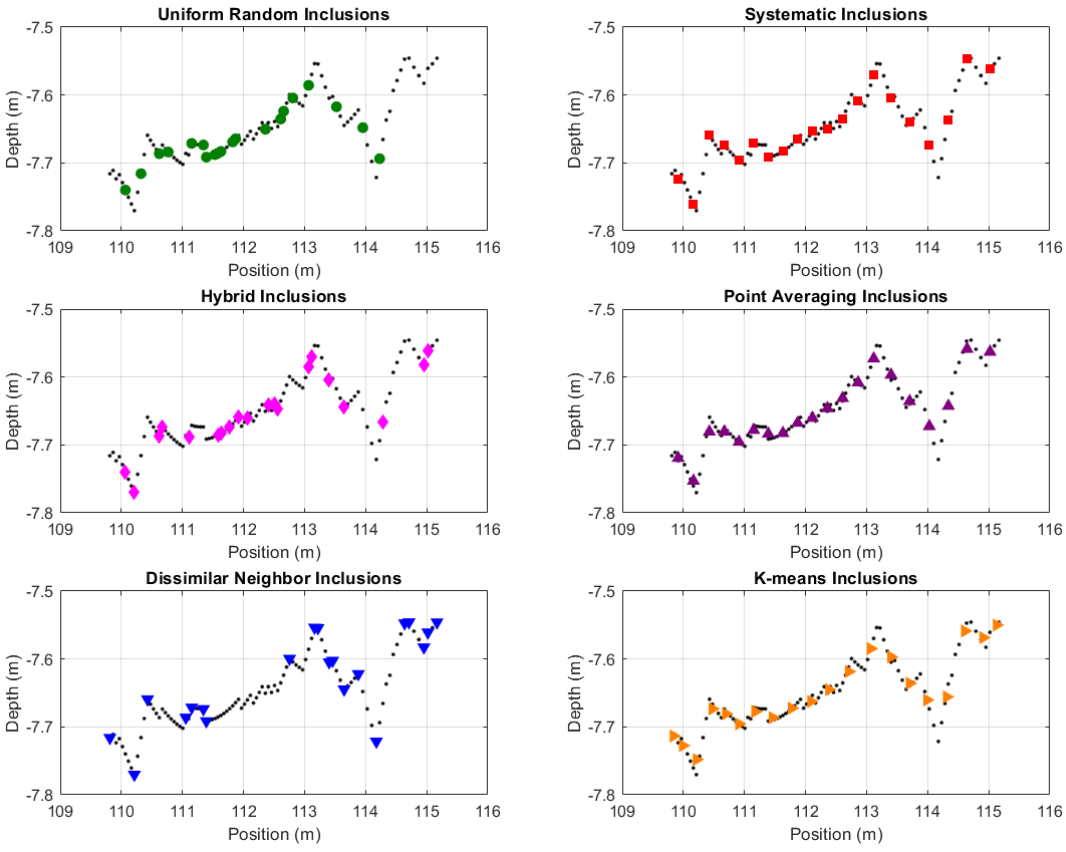

Figure 6. Inclusions (D_R) at 20% downsampling for each method. Data has 100 soundings and each method chose 20 soundings.

For consistency, each method was applied to the same sonar swath in Figure 7.

The swath exhibited desirable characteristics, including regions with and without overlap, a turn with high point density, and considerable relative depth variation (between 6m and 12m). Images of downsampled swaths for each method can be found in the Appendix.

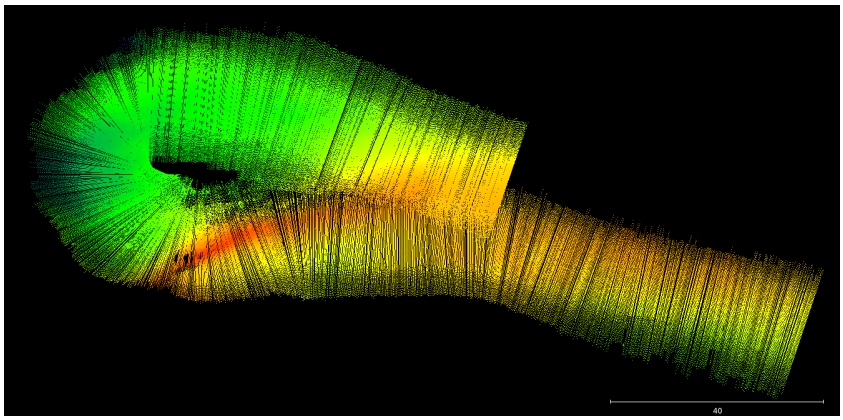

Figure 7. Top view of raw bathymetry swath used for testing, from the Wiggles Bank on the St. Mary's River in Georgia. Depth shown in color.

1.2.3 Hyperparameter Training

Optimal kernel hyperparameters $\boldsymbol{\theta} = \{l, \sigma_f^2\}$ maximize the log marginal likelihood (LML) [1]. LML rewards model fit and punishes model complexity. It is far too computationally cumbersome to calculate LML using large datasets. Thus, the training process was performed offline before running a survey. This can be done using a small patch of bathymetry data to calculate a LML heat map over ranges of l and σ_f^2 (alternatively, gradient descent can be used). The largest LML value determines the selection of $\boldsymbol{\theta}$. These parameters are inputted to MP-GPR upon starting a survey. Every variation of downsample percent s for each downsample method requires retraining $\boldsymbol{\theta}$.

Note that $\boldsymbol{\theta}$ was trained on a small 10m $\times$ 10m patch of the actual survey region. The patch contained common features that are encountered throughout

the survey. Once the hyperparameters are trained on the patch, the model applies those parameters to calculate a solution for the entire survey. Trained hyperparameter values can be found in the Appendix.

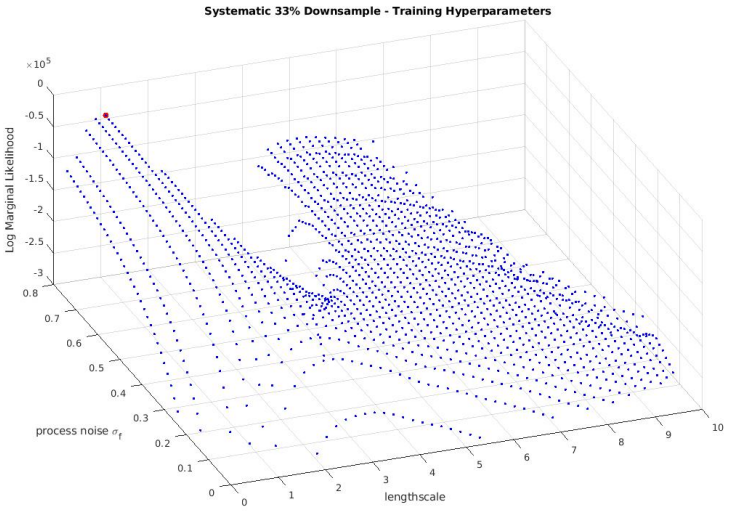

Figure 8. Log marginal likelihood plot to determine optimal hyperparameters for $s = 30$ with systematic downsampling (maxmimum value shown in red).

1.2.4 Incorporating Uncertainty

The original MP-GPR formulation assumed each training data point contains the same uncertainty value σ_n^2. Uncertainty values are stored in $\boldsymbol{W}$, the sensor noise matrix, such that $\boldsymbol{W} = \boldsymbol{I}\sigma_n^2$ where $\boldsymbol{I}$ is an $N \times N$ identity matrix. Each data point has one associated uncertainty value that represents a d-dimensional Gaussian distributed error.

Data collected by MBES are subject to multiple uncertainty sources, such as roll error, pitch error, heave error, refraction error (due to the sound speed profile), sonar system error, etc. [18]. Importantly, each data point obtained using a sonar system has unique uncertainty values in every input dimension (northing, easting, and depth).

The d-dimensional uncertainty of a sonar sounding can be visualized as a '3D uncertainty sphere' around the data point. For bathymetry data, σ_n^2 is properly expressed as an uncertainty vector $\boldsymbol{\sigma_n^2} = [\sigma_1^2, \sigma_2^2, ...\sigma_N^2]^T$ with each uncertainty value corresponding to a point in $\boldsymbol{x}$. The sensor noise matrix becomes $\boldsymbol{W} = diag(\boldsymbol{\sigma_n^2})$. Due to this formulation, capturing the dominant sources of uncertainty will include secondary error sources in other dimensions within the uncertainty sphere. Calculating $\boldsymbol{\sigma_n^2}$ must be cheap as to not subvert the computational gains from downsampling.

Capturing the dominant error in training data target values $\boldsymbol{y}$, water depth, was prioritized. The relatively largest source of depth error comes from the MBES's range and beam angle uncertainty. Outer beams in a sonar ping are subject to greater uncertainty than the inner beams. For a given beam angle ϕ_i and range r_i, uncertainty σ_i^2 can be calculated for each data point as,

$$\sigma_i^2 = \left(\frac{\partial y_i}{\partial r_i}\sigma_r\right)^2 + \left(\frac{\partial y_i}{\partial \phi_i}\sigma_\phi\right)^2 = (\cos(\phi_i)\sigma_r)^2 + (r_i sin(\phi_i)\sigma_\phi)^2$$

The range standard deviation is sonar system dependent, and the WASSP used in these surveys reported range resolution of 0.1m ($\pm$0.1m = 0.2m). Assuming a 6σ reported measurement, $\sigma_r = 0.2/6 = 0.0333$m. Beam angle resolution can be roughly estimated as total beam width divided by the number of beams. For the WASSP with a 120-degree beam width and 256 beams, $\sigma_\phi = \frac{2\pi/3}{256} = 0.0081$ radians/beam. A comparison is shown in figure 9 between the changing uncertainty vector $\boldsymbol{\sigma_n^2}$ and constant uncertainty σ_n^2.

1.3 Results

By varying downsample percent s for each downsample method, compute time and solution accuracy can be compared to the exact MP-GPR solution.

Root mean square error (RMSE) and mean absolute error (MAE) were calculated for each downsampled GPR inference with respect to exact GPR inference

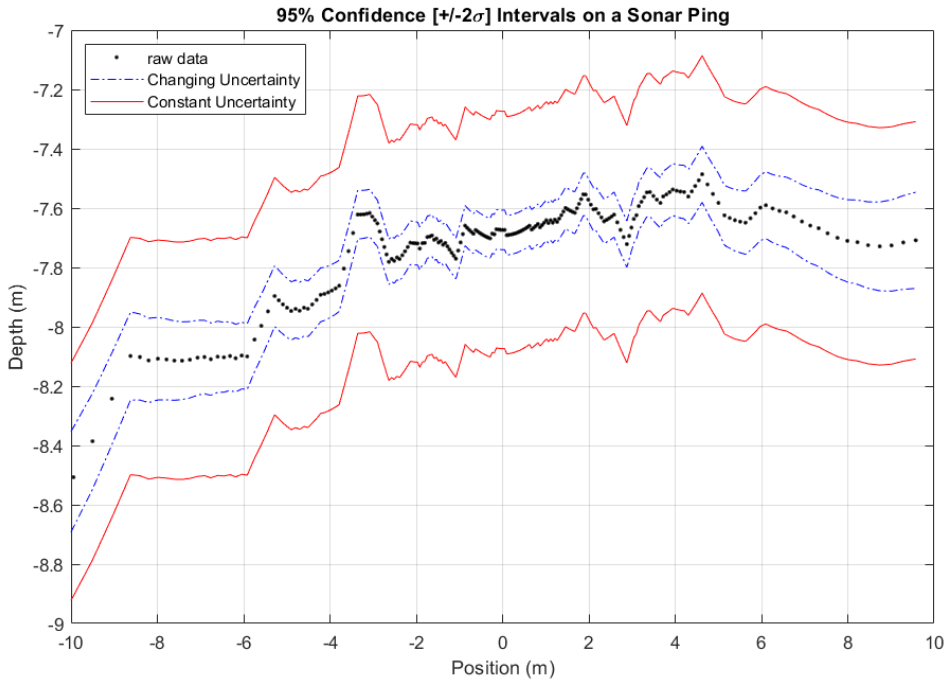

Figure 9. Uncertainty values for a given sonar ping (across track) for changing uncertainties (dashed blue) and constant uncertainties (solid red).

over all A dense inference points. All methods exhibited the trend of rapidly increasing RMSE for linearly increased downsample percent. Particularly, the order of best to worst performance from 90% to 60% downsampling was random, hybrid, K-means, and dissimilar neighbor. Systematic and average downsampling performed consistently the best from $s = 50\%$ to 25%. At the minimum $s = 10\%$, K-means performed the best and systematic performed the worst.

When running MP-GPR over the swath, hundreds of inference tiles were produced and the time to process each tile was recorded. Taking the mean of these processing tiles resulted in mean processing time as a normalized metric for method comparison.

The processing time curves for every method exhibited similar shapes, with time decreasing close to linearly with decreased s. Random, systematic, and av-

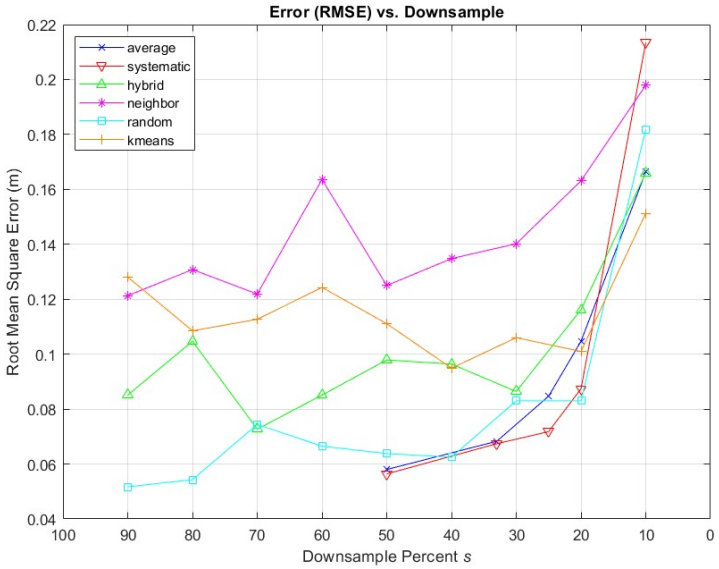

Figure 10. Root mean square error vs. downsample percent s for different methods.

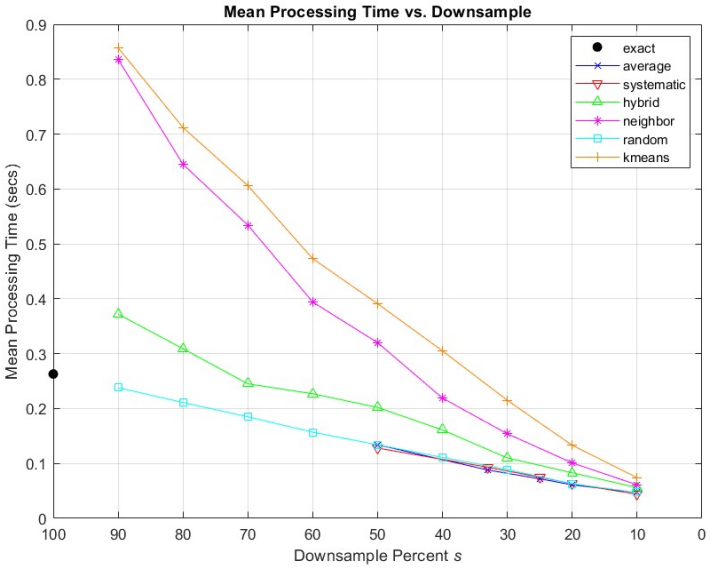

Figure 11. Mean processing time vs. downsample percent s for different methods.

erage exhibited extremely similar compute time for every downsample percent. In increasing order, hybrid, systematic, and K-means required longer compute time. When $s = 10\%$, the methods converged near a time of 0.050 seconds per tile.

1.4 Discussion

The overall trend of the results were as expected. As s decreased, every method saw increasing RMSE on average and decreasing processing time. Dissimilar neighbor's poor performance was likely due to an absence of data in the center of the sonar swath (e.g. center of the sonar swath, see image in Appendix). K-means was expected to be an improvement to random sampling and point averaging, but this was not the case. Additionally hybrid sampling failed to offer improvement over random and systematic sampling.

RMSE is largely affected by the selected points in D_R, yet chosen hyperparameters as well as random seeds may play a role. Optimizing hyperparameters was an inexact process for this data as the true maximum LML value for each trial risked divergence and was replaced by a conservative subjective estimate.

Additionally, hybrid, DN, and K-means are all slower to run than expected. As solution inference dominates overall compute time, the downsampling algorithms should not cause the magnitude of time increase present in the results. This is likely due to nuances in the MP-GPR data queueing and/or tiling algorithm, however at the time of the study the researcher did not have knowledge of this aspect of the code base.

The results indicate that for large s, it is better to use simple and cheap methods over advanced methods like K-means and dissimilar neighbor. For very small s there may be benefits to K-means and DN though the $s < 10$ solution space was not explored. In a generic MBES survey, using random, systematic, or point averaging methods would be preferred methods based on their minimal RMSE and fast compute time.

1.4.1 Future Work

An analysis of the MP-GPR queueing and tiling algorithms should be assessed to see if inconsistencies exist based off downsampling methods. This may be a contributing source to the time discrepancies seen in hybrid, DN, and K-means. They may also be easily obtainable speed-ups in the downsampling algorithms as the programmer was an engineer, not a computer scientist.

MP-GPR should be compiled on an embedded system with a GPU, such as the Nvidia Jetson Xavier, and tested to determine if real-time computation is attainable. If so, the algorithm should be deployed on an AUV/ASV for seafloor mapping and use the maximum uncertainty regions from GPR inference could as waypoints in an autonomous path-planning algorithm to re-map low confidence areas. Additionally, as any variable (not only seafloor depth) can be inputted to GPR, the algorithm is primed to map other variables such as salinity, temperature, ocean current, or other spatially distributed environmental variables.

Currently, operations such as downsampling were performed on each sonar ping. While this approach aligns with processing data as it obtained, expanding the operations to run over multiple pings gathered together over time may show improvements. This could cause along track and across track data density to be similar, improving GPR training and inference. This approach would also enable calculation and comparison of 3D surface normals to contribute to the uncertainty analysis.

Alternative kernels should be explored. Particularly, the Matern kernel trains unique length scales for each dimension of the data could improve GPR inference for data with variable across track and along spacing as seen in this study. Matern kernels also have a smoothness parameter ν which may offer a smoother seafloor curvature fit.

An efficient, intelligent hyperparameter trainer would provide consistent means to train parameters. This would also contribute to live, adaptive hyperparameters that can be retrained during a survey. Adaptive parameters would allow MP-GPR to accurately map environments as they change (i.e. from sand ripples to shipwreck). However, the high compute cost of the log marginal likelihood is prohibitive.

The uncertainty vector calculations could be paired with a higher fidelity uncertainty model, such as the Combined Uncertainty and Bathymetric Estimator (CUBE) [19]. CUBE a statistical processing algorithm for multi-beam sonar that would be a useful tool for applications that require precisely-bounded uncertainty estimates at each sounding.

1.5 Conclusion

This study built upon the in-situ MP-GPR mapping algorithm by introducing multiple downsampling methods to reduce required compute time. A preprocessing filter was applied to remove outliers as to not bias certain methods from selecting outliers as 'interesting points.' For each downsampled data point, an uncertainty value was assigned using by combining the beam angle and range uncertainty along the seafloor. Optimal hyperparameters were trained offline on a sub-swath of data. During live simulations, raw pings were streamed in, filtered, downsampled, assigned uncertainty values, queued together into blocks. Inference was performed on blocks by periodically generate on a dense grid of point cloud tiles to form a 2.5D model of the seafloor.

The downsampling methods explored were random, systematic, hybrid, point averaging, dissimilar neighbor, and K-means. Decreasing the downsample percent led to a reduction in computation time for each method, but also resulted in an increase in RMSE. The best methods were systematic (decimation) and point

averaging, with random downsampling as a reasonable alternative. Hybrid, K-means, and dissimilar neighbor under performed with long compute times as well as high RMSE values.

LIST OF REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[2] S. Barkby, S. B. Williams, O. Pizarro, and M. V. Jakuba, "Bathymetric particle filter slam using trajectory maps," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1409–1430, 2012.

[3] T. Hitchcox and J. R. Forbes, "A point cloud registration pipeline using gaussian process regression for bathymetric slam*," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4615–4622, 2020.

[4] T. Ma, Y. Li, R. Wang, Z. Cong, and Y. Gong, "Auv robust bathymetric simultaneous localization and mapping," *Ocean Engineering*, vol. 166, pp. 336–349, 2018.

[5] K. Krasnosky and C. Roman, "A massively parallel implementation of gaussian process regression for real time bathymetric modeling and simultaneous localization and mapping," *Field Robotics*, vol. 2, pp. 940–970, 03 2022.

[6] M. Franey, P. Ranjan, and H. Chipman, "A short note on gaussian process modeling for large datasets using graphics processing units," 2012.

[7] R. B. Gramacy, J. Niemi, and R. Weiss, "Massively parallel approximate gaussian process regression," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 2, pp. 564–548, 2013.

[8] A. Melkumyan and F. Ramos, "A sparse covariance function for exact gaussian process inference in large datasets." in *IJCAI*, vol. 9, 2009, pp. 1936–1942.

[9] B. W. Silverman, "Some aspects of the spline smoothing approach to non-parametric regression curve fitting," *Royal Statistical Society, Series B*, vol. 47, 1985.

[10] N. D. Lawrence, M. W. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," in *NIPS*, 2002.

[11] M. W. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *International Conference on Artificial Intelligence and Statistics*, 2003.

[12] M. K. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *International Conference on Artificial Intelligence and Statistics*, 2009.

[13] "Robotic operating system (ros)," https://www.ros.org/, accessed: 2021-09-30.

[14] "Cuda runtime api documentation."

[15] K. Krasnosky, C. Roman, and D. Casagrande, "A bathymetric mapping and slam dataset with high-precision ground truth for marine robotics," *The International Journal of Robotics Research*, vol. 41, no. 1, pp. 12–19, 2022.

[16] M. R. Anderberg, *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*. Academic press, 2014, vol. 19.

[17] J. Peña, J. Lozano, and P. Larrañaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1027–1040, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865599000690

[18] R. Hare, "Depth and position error budgets for mulitbeam echosounding," *International Hydrographic Review*, vol. 72, 1995.

[19] B. Calder, "Automatic statistical processing of multibeam echosounder data," *The International Hydrographic Review*, vol. 4, no. 1, Jan. 2003. [Online]. Available: https://journals.lib.unb.ca/index.php/ihr/article/view/20603

APPENDIX A

Notation and Relevant Equations

The following variables and parameters are present in GPR equations. Bold symbols indicate matrix quantities and standard roman characters indicate scalar values.

- D is the set of training data, consisting of input vectors mapping to target scalars, which is the input to a GPR model, $D = \{(\boldsymbol{x}_i, y_i)|i = 1, \ldots, N\}$
- N is the number of data points in the training data
- d is the number of dimensions of the input data $\boldsymbol{x}$
- $\boldsymbol{x}$ is a $N \times d$ matrix of input values and the independent variables in the training data, $\boldsymbol{x_i} = [x_1, x_2, ..., x_d]$
- $\boldsymbol{y}$ is a $N \times 1$ matrix of target values and part of the training data
- f is the underlying function of interest that maps $\boldsymbol{x}_i$ to y_i, such that $y_i = f(\boldsymbol{x}_i) + \epsilon$
- ϵ is the observation error assumed to be independent identically distributed Gaussian noise with $\mu = 0$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$
- σ_n^2 is the scalar sensor noise variance
- $\boldsymbol{\sigma_n^2}$ is the sensor noise vector, $\boldsymbol{\sigma_n^2} = [\sigma_1^2, \sigma_2^2, ..., \sigma_N^2]^T$
- A is the number of prediction points for inference
- $\boldsymbol{x}_*$ is a $A \times d$ matrix of prediction input points for inference
- $\boldsymbol{y}_*$ is a $A \times 1$ matrix of the solution means of GPR (output prediction points determined by inference with the GPR model using $\boldsymbol{x}_*$)
- $\boldsymbol{\Sigma}_*$ is a $A \times A$ output covariance matrix with diagonal values as the solution variances
- $k(\boldsymbol{x}, \boldsymbol{x'})$ is the kernel function used to calculate covariances

- l is the lengthscale present in the kernel function as a hyperparameter
- σ_f^2 is the process noise (variance) of the underlying function f and is present in the kernel function as a hyperparameter
- $\boldsymbol{\theta}$ is the set of hyperparameters in the kernel function, $\boldsymbol{\theta} = \{l, \sigma_f^2\}$
- s is the downsample percent parameter, $0 \leq s \leq 1$
- $\boldsymbol{K}$ is a $N \times N$ auto-covariance matrix for input points $\boldsymbol{x}$, $\boldsymbol{K} = k(\boldsymbol{x}, \boldsymbol{x})$
- $\boldsymbol{K}_*$ is a $A \times N$ covariance matrix for prediction input points $\boldsymbol{x_*}$ and input points $\boldsymbol{x}$, $\boldsymbol{K_*} = k(\boldsymbol{x_*}, \boldsymbol{x})$
- $\boldsymbol{K}_{**}$ is a $A \times A$ covariance matrix for prediction input points $\boldsymbol{x}_*$, $\boldsymbol{K}_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$
- $\boldsymbol{W}$ is a $N \times N$ sensor noise matrix with variances pertaining to each input point. For constant variances, $\boldsymbol{W} = \sigma_n^2 \boldsymbol{I}$. For datapoints with changing variances, $\boldsymbol{W} = diag(\boldsymbol{\sigma_n^2})$.
- $\boldsymbol{I}$ is the identity matrix
- $\boldsymbol{V}$ is a $N \times N$ complete input covariance matrix, $\boldsymbol{V} = \boldsymbol{K} + \boldsymbol{W}$

The standard equations that govern GPR with a squared exponential kernel are:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 exp(-\frac{|\boldsymbol{x} - \boldsymbol{x'}|^2}{2l^2}) \tag{A.1}$$

$$\boldsymbol{y}_* = \boldsymbol{K}_* \cdot \boldsymbol{V}^{-1} \cdot \boldsymbol{y} \tag{A.2}$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_* \cdot \boldsymbol{V}^{-1} \cdot \boldsymbol{K}_*^T \tag{A.3}$$

MP-GPR uses Cholesky decompositions for speed and to ensure solution convergence and utilizes a modified version of these standard equations. The motivation and equations are described in great detail by Krasnosky [5].

The approximation to the squared exponential kernel used in MP-GPR is:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \begin{cases} \sigma_f^2[\frac{2+cos(2\pi\frac{d}{l})}{3}(1-\frac{d}{l}) + \frac{1}{2\pi}sin(2\pi\frac{d}{l})] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases} \tag{A.4}$$

The log marginal likelihood (LML) is defined as:

$$LML = -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{V}^{-1}\boldsymbol{y} - \frac{1}{2}log|\boldsymbol{V}| - \frac{N}{2}log(2\pi) \tag{A.5}$$

Equations relating to sonar uncertainty calculations are:

$$y_i = r_i\cos(\phi_i)$$
$$\sigma_i^2 = \sigma_{d_i}^2 = \left(\frac{\partial y_i}{\partial r_i}\sigma_r\right)^2 + \left(\frac{\partial y_i}{\partial \phi_i}\sigma_\phi\right)^2$$
$$\sigma_i^2 = \left(\cos(\phi_i)\sigma_r\right)^2 + \left(r_i sin(\phi_i)\sigma_\phi\right)^2$$

The likelihood that an approximate solution is equal to an exact solution as a given inference point $\boldsymbol{x_i}$ is given:

$$pointLikelihood = p(z_{approx} - z_{exact} = 0) = \frac{\exp -\frac{1}{2}\frac{(\mu_{z_{approx}} - \mu_{z_{exact}})^2}{\sigma_{z_{approx}}^2 + \sigma_{z_{exact}}^2}}{\sqrt{2\pi(\sigma_{z_{approx}}^2 + \sigma_{z_{exact}}^2)}} \tag{A.6}$$

The formula for root mean square error is:

$$RMSE = \sqrt{\sum_{i=1}^{A}\frac{(y_{*_{exact}}^i - y_{*_{approx}}^i)^2}{A}} \tag{A.7}$$

The formula for mean absolute error is:

$$MAE = \frac{1}{A}\sum_{i=1}^{A}(y_{*_{exact}}^i - y_{*_{approx}}^i) \tag{A.8}$$

MP-GPR uses Cholesky decompositions for speed and convergence. Those equations are described in great detail by Krasnosky [5].

APPENDIX B

Significant Parameters

This section lists the compute time and trained hyperparameters and for each downsampling method. Compute time is the total compute time (downsample time plus inference time) divided by the number of inference tiles computed. An inference tile is the output of MP-GPR and a dense point cloud ($\boldsymbol{x}_* = \{northing, easting\}, \boldsymbol{y}_* = \{depth\}$).

s	Random	Hybrid	DN	K-means	Systematic	Average
90%	0.238	0.371	0.836	0.852	-	-
80%	0.211	0.308	0.645	0.708	-	-
70%	0.185	0.244	0.534	0.602	-	-
60%	0.157	0.226	0.394	0.470	-	-
50%	0.134	0.201	0.320	0.389	0.128	0.134
40%	0.111	0.160	0.219	0.303	-	-
33%	-	-	-	-	0.093	0.088
30%	0.089	0.110	0.154	0.214	-	-
25%	-	-	-	-	0.075	0.072
20%	0.063	0.083	0.101	0.132	0.062	0.061
10%	0.047	0.055	0.061	0.074	0.044	0.047

Table B.1. Compute Times (seconds) for Downsampling Methods

s	RMSE	MAE	l	σ_f	Compute Time
90%	0.0517	0.0319	1.06	0.09	0.238
80%	0.0543	0.0346	1.06	0.11	0.211
70%	0.0744	0.0408	1.06	0.15	0.185
60%	0.0665	0.0370	1.06	0.15	0.157
50%	0.0638	0.0385	1.06	0.19	0.134
40%	0.0625	0.0384	1.06	0.24	0.111
30%	0.0830	0.0446	1.27	0.25	0.089
20%	0.0830	0.0477	1.27	0.39	0.063
10%	0.1819	0.0849	1.45	0.60	0.047

Table B.2. Results and Parameters for Random Downsampling

s	RMSE	MAE	l	σ_f	Compute Time
90%	0.0851	0.0522	1.06	0.11	0.371
80%	0.1047	0.0595	1.06	0.11	0.308
70%	0.0727	0.0436	1.06	0.11	0.244
60%	0.0851	0.0509	1.06	0.15	0.226
50%	0.0979	0.0576	1.08	0.19	0.201
40%	0.0964	0.0586	1.08	0.23	0.160
30%	0.0864	0.0526	1.08	0.30	0.110
20%	0.1162	0.0676	1.08	0.50	0.083
10%	0.1659	0.1009	1.18	0.62	0.055

Table B.3. Results and Parameters for Hybrid Downsampling

s	RMSE	MAE	l	σ_f	Compute Time
90%	0.1213	0.0723	1.25	0.09	0.836
80%	0.1308	0.0814	1.06	0.09	0.645
70%	0.1219	0.0749	1.06	0.11	0.534
60%	0.1636	0.0919	1.06	0.13	0.394
50%	0.1250	0.0728	1.06	0.21	0.320
40%	0.1348	0.0828	0.96	0.31	0.219
30%	0.1402	0.0917	0.96	0.39	0.154
20%	0.1631	0.1094	0.96	0.49	0.101
10%	0.1979	0.1376	0.96	0.71	0.061

Table B.4. Results and Parameters for Dissimilar Neighbor Downsampling

s	RMSE	MAE	l	σ_f	Compute Time
90%	0.1281	0.0729	0.88	0.13	0.852
80%	0.1085	0.0619	1.06	0.17	0.708
70%	0.1127	0.0632	1.06	0.17	0.602
60%	0.1243	0.0709	1.06	0.17	0.470
50%	0.1111	0.0635	1.14	0.19	0.389
40%	0.0949	0.0565	1.23	0.22	0.303
30%	0.1060	0.0624	1.05	0.39	0.214
20%	0.1010	0.0626	1.01	0.70	0.132
10%	0.1513	0.0939	1.01	1.64	0.074

Table B.5. Results and Parameters for K-means Downsampling

s	RMSE	MAE	l	σ_f	Compute Time
50%	0.0564	0.0362	1.06	0.18	0.128
33%	0.0674	0.0393	1.06	0.27	0.093
25%	0.0718	0.0422	1.06	0.36	0.075
20%	0.0874	0.0512	1.07	0.62	0.062
10%	0.2134	0.1034	1.01	1.12	0.044

Table B.6. Results and Parameters for Systematic Downsampling

s	RMSE	MAE	l	σ_f	Compute Time
50%	0.0580	0.0355	1.08	0.16	0.134
33%	0.0683	0.0412	1.08	0.28	0.088
25%	0.0847	0.0460	1.08	0.39	0.072
20%	0.1047	0.0531	1.08	0.46	0.061
10%	0.1663	0.0953	1.11	1.60	0.047

Table B.7. Results and Parameters for Average Downsampling

APPENDIX C

Downsampled Swaths

The following swaths show the bathymetry used. The raw bathymetry contains all data streamed in from the MBES. The downsampled swaths show all methods for $s = 20\%$. The raw swath was 150m wide and 60m tall with 330,000 soundings.

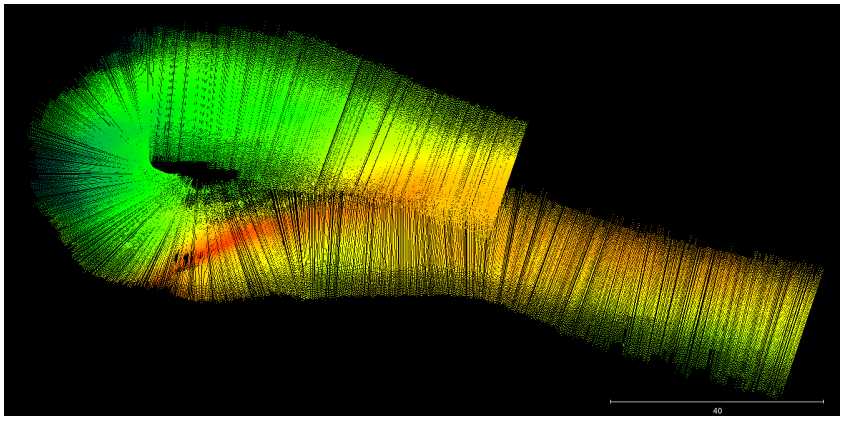

Figure C.1. Top view of the raw bathymetry.

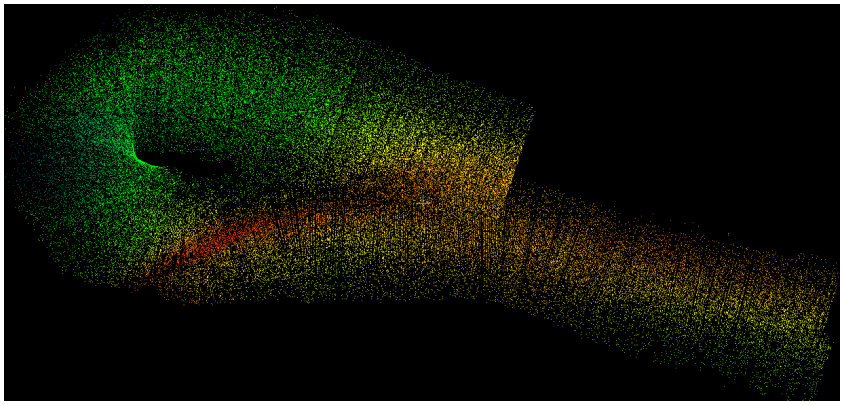

Figure C.2. Top view of the 20% random downsampled swath.

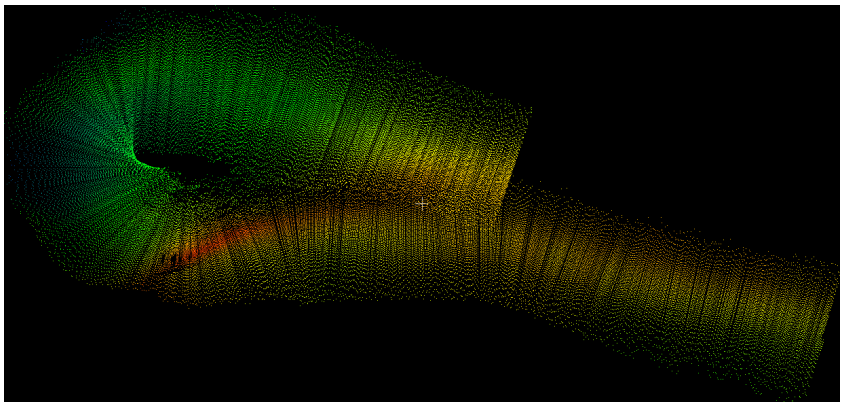

Figure C.3. Top view of the 20% systematic downsampled swath.

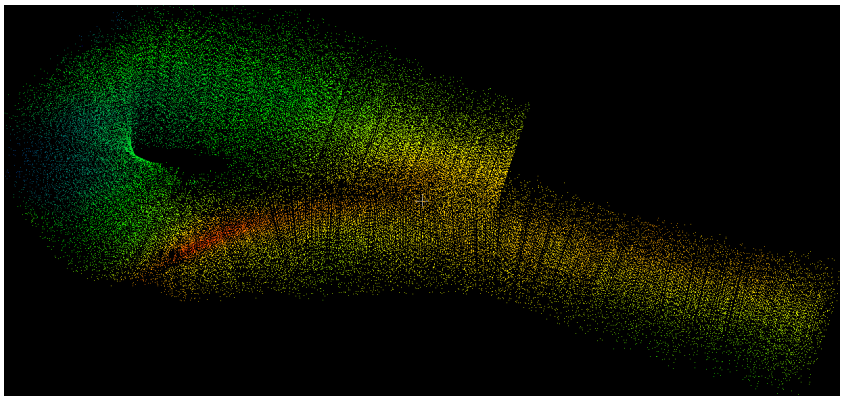

Figure C.4. Top view of the 20% hybrid downsampled swath.

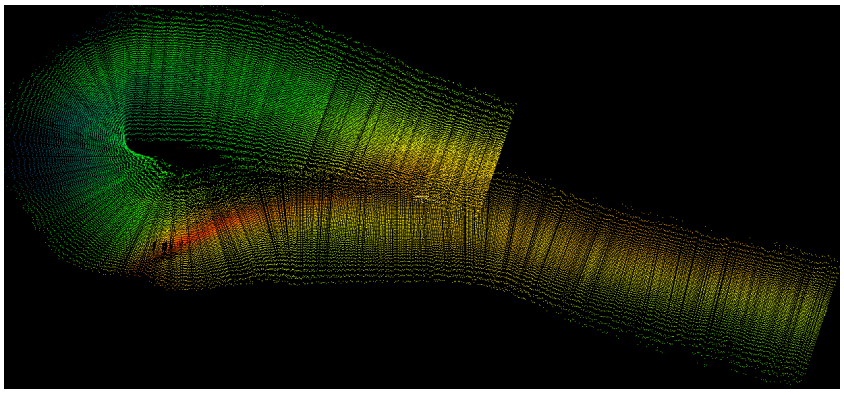

Figure C.5. Top view of the 20% average downsampled swath.

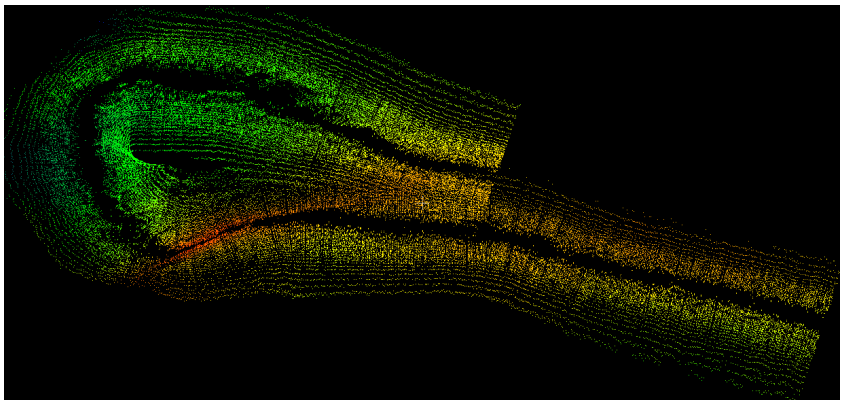

Figure C.6. Top view of the 20% dissimilar neighbor downsampled swath.

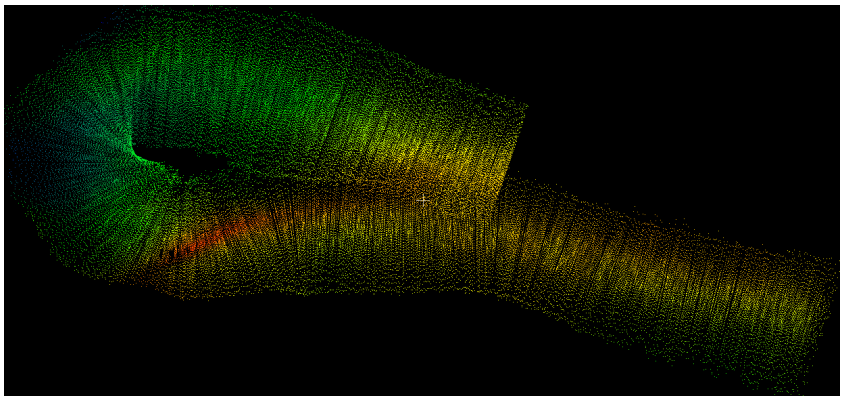

Figure C.7. Top view of the 20% K-means downsampled swath.