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

In this paper we further the capabilities of Massively Parallel Gaussian Process Regression (GPR) in two ways. The first introduces data reduction techniques to minimize inputs to the GPR algorithm while maintaining solution accuracy above a threshold. Various approaches are analyzed to find an optimal data minimization strategy. The second informs uncertainty predictions by considering the physics of multi-beam sonar refraction throughout the water column. In tandem, the presented methodologies decrease the GPR algorithm run-time and increase the fidelity of uncertainty predictions. The program requires a Graphical Processing Unit (GPU) to run and is implemented using the Robot Operating System (ROS) with C++. The algorithm was performed on pre-recorded ROS data (ROSbag) of a field survey conducted using a WASSP multi-beam sonar system at the Wiggles Bank on St. Mary's River in Georgia, USA. Results show... INPUT SENTENCE HERE DESCRIBING TAKEAWAYS.

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 4

1.2 Implementation 5

1.2.1 System Overview 6

1.2.2 Downsampling 7

1.2.3 Hyperparameter Training 10

1.2.4 Incorporating Uncertainty 11

1.3 Results 12

1.4 Discussion 12

1.4.1 Future Work 13

1.5 Conclusions 15

Page

LIST OF REFERENCES 16

APPENDIX

A Notation and GPR Equations 18

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 can contain upwards of one hundred datapoints and surveys that ping the seafloor for hours will result in millions of datapoints. 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 GPR (MP-GPR) 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 and $y_i = \text{depth}$ is the corresponding depth at that location. 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 $\boldsymbol{y}^*$ over a new set of geographically dense inputs $\boldsymbol{x}^*$ to form a point cloud map.

An important component of any GPR model is the kernel which describes the relevant lengths scales and variance 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 $\boldsymbol{x'}$, and $\boldsymbol{\theta} = \{l, \sigma_f^2\}$ are the tunable hyperparameters. The full set of GPR equations are provided in the Appendix.

1.1.2 Massively Parallel Gaussian Process Regression

The underlying computation of a GPR suffers from mathematical inefficiencies when training on large datasets, primarily due to the inverse operation of a $N \times N$ covariance matrix which requires $O(N^3)$ operations. This renders the basic implementation of a GPR intractable for massive data volumes [1]. Multi-beam sonar surveys often generate millions of datapoints and hence cannot be computed using basic GPR.

Krasnosky's Massively Parallel GPR (MP-GPR) introduced a parallel computational framework to speed up the computation. MP-GPR proposed an updateable GPR model that leveraged parallel processor cores on a GPU, iterative matrix updates that incorporate new bathymetric data as it is collected, 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]. These modifications allow for a tractable, online, and exact GPR solution that generates a terrain model with uncertainties from a stream of input points during a multi-beam survey. MP-GPR can operate in real-time but requires significant computer 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 for 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 would diverge from MP-GPR's implementation. However, data-centric methods incorporate techniques to replace training data D with a fabricated a 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 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 with MP-GPR results in a fast 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. Downsampling methods were then implemented and the hyperparameters were retrained. To test each method multiple data-processing runs were conducted to determine the average compute time and error statistics were calculated to measure the differences between each approximate solution and the exact MP-GPR solution.

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 is well mixed and measured sound speed profile nearly constant with depth, alleviating

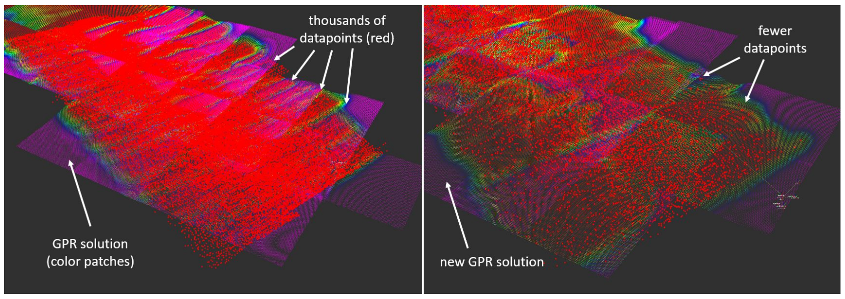

Figure 1. Snapshot of ROS Rviz simulation running MP-GPR with all N points (left) and with 10% random downsampling (right). Individual soundings (datapoints) are shown in red. GPR solution tiles are shown in color.

the need for ray tracing corrections. The mapping system and survey details are fully explained by Krasnosky, et. al [15].

1.2.1 System Overview

The processing workflow (2) is setup to ingest 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 and downsamples the data. The filtered sonar pings are queued into tiles. GPR inference is performed on the tiles to generate predictive points with uncertainties, which are added to the bathymetry map. This is a continuous process throughout a survey [5].

The downsampling methods considered are:

- 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

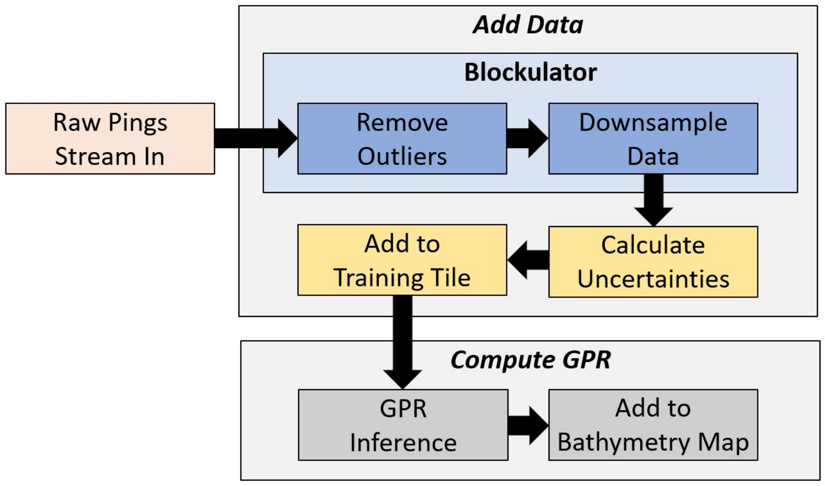

Figure 2. Block diagram showing MP-GPR data flow.

points,

- a *Dissimilar Neighbor* method that uses a ratio of 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 as a pre-processing step as live data 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) without consideration of any previously collected data. Downsampled pings are queued into blocks which are processed periodically and added to the GPR solution. The resulting time complexity is approximately $O(M^3)$.

As some downsampling methods select the most 'unique' values from a set of points, outliers may be included more often than real data. 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 2.25x the standard deviation of the neighboring points' mean (excluding p). Outliers were removed from the set, and the clean set was passed to the following downsampling algorithms.

Uniform Random

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

Systematic Random

Systematic random sampling (”decimation”) sorts the data by a predetermined ‘ordered variable’ and selects every jth point along that dimension. Sonar data be systematically selected using ordered indices (e.g. in MATLAB ($x_o : j : x_{end}$)). To avoid along track selection bias, the first index of each ping x_0 was uniform randomly selected and subsequent points were systematically selected.

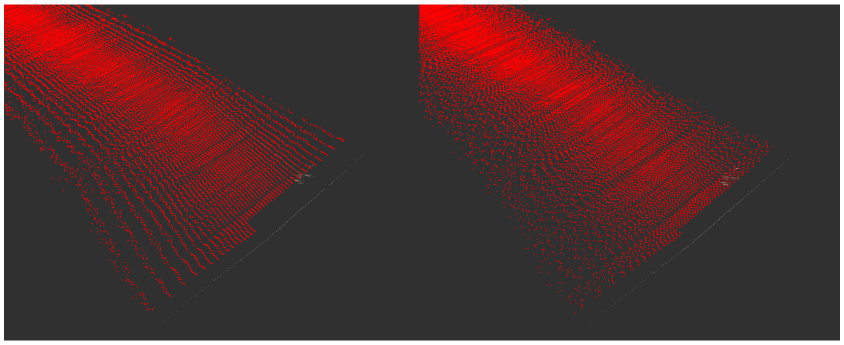

Figure 3. 20% systematic downsampling with biased (left) and unbiased (right) approaches

Hybrid Random

The hybrid random method recognizes the value in both random and systematic sampling. 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 datapoints to create a new representative datapoint. In this manner, information from all data points influences the solution.

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 their neighbors [16].

In the spirit of seeking 'unique' points to represent the distribution of a sonar ping, similar neighbor checks the two neighboring points of p_i, 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 along a ping vary.

K-means

K-means is a popular clustering algorithm that splits a group of samples into k-clusters (subgroups) based on their similarity [17]. To combat performance issues [17] and keep processing time minimal, every jth point is selected as the initial centroids and the number of iterations are limited. The centroids of k-clusters serve as the new datapoints.

Downsampling Implementation

Each method has one adjustable parameter: *downsample percent*, s. A small s value drastically reduces compute time and decreases solution accuracy, and vice versa. s was systematically varied for each method to determine the optimal value that minimizes compute time while maximizing solution accuracy (above 98%).

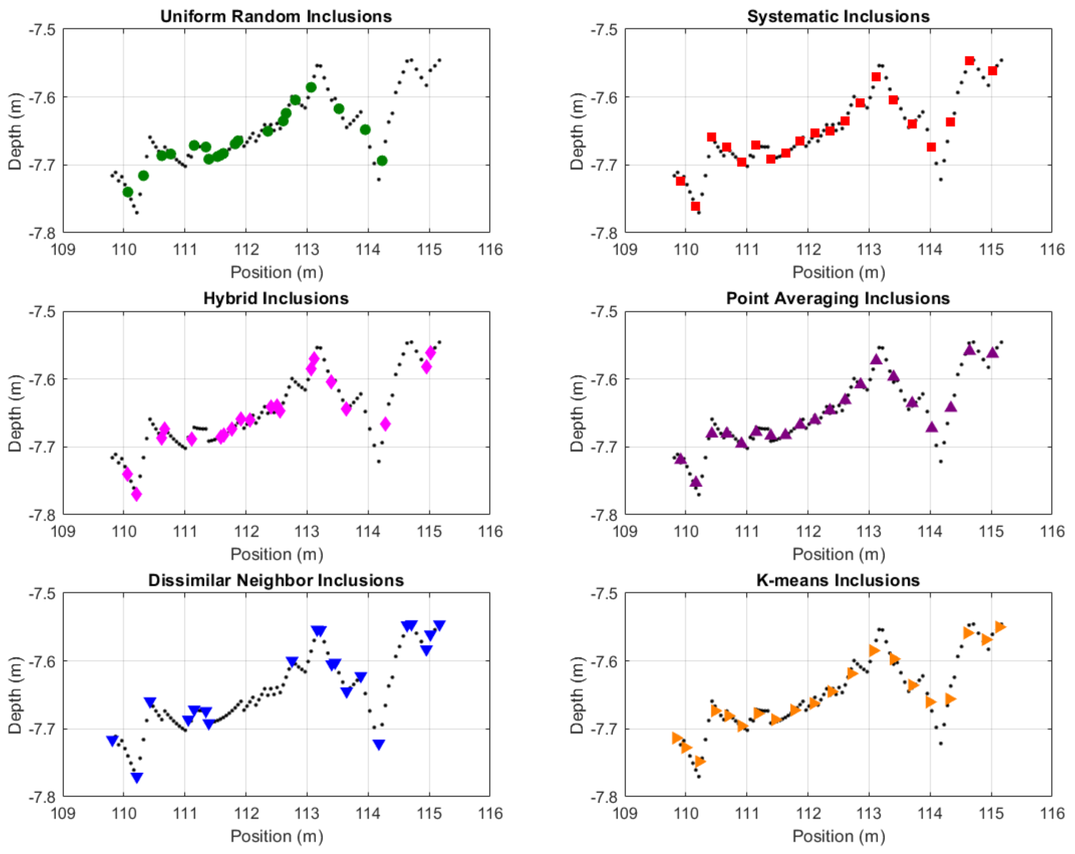

Figure 4. Inclusions (D_R) at 20% downsampling for each method. Data has 100 soundings of a sonar ping and each method has 20 soundings.

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 $\boldsymbol{\theta}$ is trained on a small portion of the actual survey region. The patch should contain 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.

1.2.4 Incorporating Uncertainty

MP-GPR assumes each training datapoint contains the same uncertainty value σ_n^2. Uncertainty values are stored in 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 datapoint 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).

For a given data point, GPR expresses d-dimensional uncertainties with a single value which can be visualized as a '3D uncertainty sphere' around the data point. 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 error in training data target values $\boldsymbol{y}$, water depth, was prioritized. A dominant 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, the depth y_i and uncertainty σ_i^2 for each data point can be can be calculated as,

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

The range standard deviation is sonar system dependent, and the WASSP used in these surveys reported range resolution of 0.1m ($\pm$0.05m). Assuming a 6σ reported measurement, $\sigma_r = 0.1/6 = 0.0167$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.

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.

WHICH METHOD PERFORMED BEST?

1.4 Discussion

INCOMPLETE

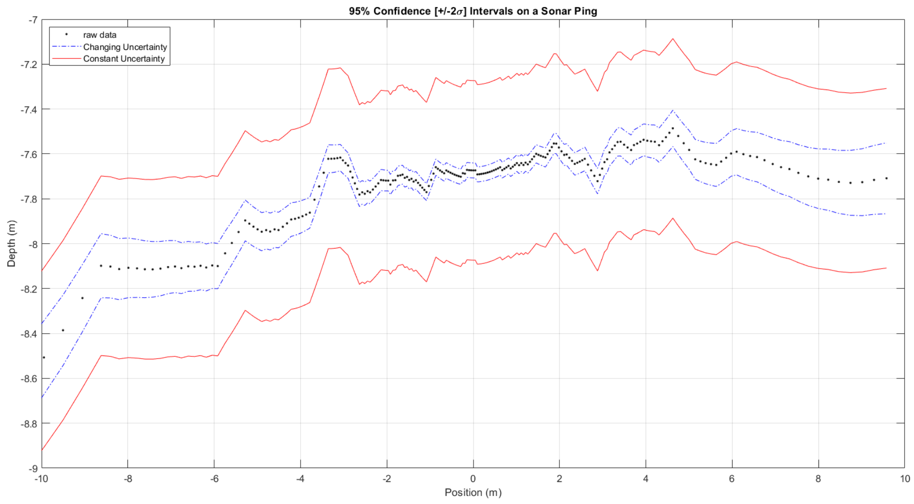

Figure 5. Uncertainty values for a given sonar ping (across track) for changing uncertainties (dashed blue) and constant uncertainties (solid red).

Science
coming
soon

Figure 6. Compute time vs. downsample percent for different approximate methods

1.4.1 Future Work

Currently, operations such as downsampling were performed on each sonar ping. Expanding the operations to run over tiles (multiple pings gathered together

Science
coming
soon

Figure 7. Table showing accuracy, compute time, and downsample percent for different approximate methods

over time) may show improvements. This could cause along track and across track data density to be similar, improving GPR inference. This tile-by-tile approach would also enable calculation and comparison of 3D surface normals to contribute to the uncertainty analysis. CUBE [19], a statistical processing algorithm for multibeam sonar, could be integrated into MP-GPR to provide uncertainty estimates at each sounding as well.

A logical next step is preparing MP-GPR for field work. MP-GPR should be compiled on an embedded system with a GPU, such as the Nvidia Jetson Xavier, and deployed on an AUV/ASV for seafloor mapping. The maximum uncertainty points from GPR inference could be passed to an autonomous path-planning algorithm for re-mapping of low confidence areas. As any variables (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.

1.5 Conclusions

INCOMPLETE This thesis builds upon the MP-GPR algorithm by introducing multiple downsampling methods to reduce required compute time and by accounting for uncertainty values of each sonar datapoint.[1]

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," 2003.

APPENDIX A

Notation and GPR Equations

The following variables and parameters are present in GPR equations. Bold symbols indicate matrix quantities, 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 $\boldsymbol{x}$ data
- $\boldsymbol{x}$ 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 unique 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 equations that govern GPR with a squared exponential kernel are:

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

$$\boldsymbol{y}_* = \boldsymbol{K}_* \cdot \boldsymbol{V}^{-1} \cdot \boldsymbol{y}$$

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

MP-GPR utilizes an approximation to the squared exponential kernel:

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

The log marginal likelihood (LML) is defined as:

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