December 2021

Dynamic Reduction of Scientific Data Through Spatiotemporal Properties

Megan Louise Hickman Fulp
Clemson University, mhickman828@gmail.com

Dynamic Reduction of Scientific Data Through Spatiotemporal Properties

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Megan Hickman Fulp
December 2021

Accepted by:
Dr. Jon C. Calhoun, Committee Chair
Dr. Melissa C. Smith
Dr. Walt Ligon
Dr. Rong Ge

Abstract

Improvements in High-Performance Computing (HPC) has enabled researchers to develop more sophisticated simulations and applications which solve previously intractable problems. While these applications are critical to scientific innovation, they continue to generate even larger quantities of data, which only worsens the existing I/O bottleneck. To resolve this issue, researchers use various forms of data reduction.

Currently, researchers have access to many different types of data reduction. These include methods such as data compression, time-step selection, and data sampling. While each of these are effective methods, data compression algorithms and data sampling methods do not leverage the temporal aspect of the data, and time-step selection is prone to missing critical abrupt changes. With this in mind, we develop our spatiotemporal data sampling method.

In this thesis, we develop a spatiotemporal data sampling method that leverages both the spatial and temporal properties of simulation data. Specifically, our method compares corresponding regions of the current time-step with that of the previous time-step to determine whether data from the previous time-step is similar enough to reuse. Additionally, this method biases more rare data values during the sampling process to ensure regions of interest are kept with higher fidelity. By operating in this manner, our method improves sample budget utilization and, as a result, post-reconstruction data quality. As the effectiveness of our method relies heavily on user input parameters, we also provide a set of pre-processing steps to alleviate the burden on the user to set appropriate ones. Specifically, these pre-processing steps assist users in determining an optimal value for the number of bins, error threshold, and the number of regions. Finally, we demonstrate the modularity of our sampling process by demonstrating how it works with any different internal core sampling algorithm.

Upon evaluating our spatiotemporal sampling algorithm, we find it is capable of achieving

higher post-reconstruction quality than Biswas et al.'s non-reuse importance-based sampling method. Specifically, we find our method achieves a 31.3% higher post-reconstruction quality while only introducing a 37% degradation in throughput, on average. When assessing our pre-processing steps, we find they are efficient at assisting users in determining an optimal value for the number of bins, error threshold, and the number of regions. Finally, we illustrate the modularity of our sampling method by showing how one would swap the core sampling algorithm. From our evaluation, we find our spatiotemporal sampling method is an effective choice for sampling simulation data.

Acknowledgments

I first and foremost would like to thank my committee chair, Dr. Jon C. Calhoun, for his guidance and encouragement throughout my time at Clemson. I would also like to give thanks to Ayan Biswas with Los Alamos National Laboratory, who introduced me to the field of data sampling and served as a great mentor. I also thank Dr. William Jones and Dr. Nathan DeBardeleben, who first introduced me to scientific research. Finally, I am grateful to Dr. Melissa Smith, Dr. Walt Ligon, and Dr. Rong Ge for serving as part of my committee.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

High-performance computing (HPC) systems are capable of performing complex calculations and processing data at high speeds, with today's systems capable of performing 442,010 trillion floating-point operations per second [1]. Researchers from a plethora of disciplines use this tremendous computational power to solve previously intractable complex problems. Some examples of these problems include studying the universe's evolution, climate change, storms, artificial intelligence, and medical research. Virtually all scientific research fields have the ability to leverage the capabilities of HPC systems.

As the capabilities of HPC systems advance, so does the amount of data HPC applications produce. From 2008 to 2018, Cappello et al. found that the computational power of HPC systems has increased by 114× [8]. Researchers leverage this extra computational power to run more complex scientific simulations. For instance, the Coupled Model Intercomparison Project Phase 5 (CMIP5) [34] is an international project that consists of research centers around the world running the same set of simulations to study climate and other problems. One such simulation is the Community Earth System Model [14] (CESM) which simulates various climate events around the world.

While these complex simulations are necessary, they also produce an increasing amount of data, making processing and storing the data more challenging. This issue is easily seen when using the CESM simulation. First, Baker et al. detail how a single modern high-resolution CESM simulation can generate one half a terabyte of data per simulation year [4]. Additionally, Mickelson et al. found that the CESM simulation generated 2.5 PB of data over the course of 18 months, but an additional 18 months was needed to post-process and publish only 6.6% of the data [25]. The

extensive time required to post-process the data is partly due to the strenuous process of storing and loading these large quantities of data. Performing I/O operations on large datasets is difficult due to disparities in HPC system capabilities. While Cappello et al. found the computational power increased 114× over the ten years, they also found the I/O bandwidth only improved 10.4× over the same period [8]. Due to the lack of I/O capabilities, working with large quantities of data creates a bottleneck in the overall HPC workflow.

Previous works have aimed to alleviate this bottleneck by reducing the overall volume of data before transferring it. While traditional data reduction methods, such as data deduplication [24] and lossless compression, do reduce data with no loss in precision, they suffer from limited reduction ratios on scientific floating-point data. Such methods achieve limited ratios due to the high entropy in the IEEE 754 floating-point mantissa bits that most HPC applications use. This limitation is an issue for the even larger data files that will occur in the next phase of CMIP (CMIP6) [11]. These files will require better levels of reduction to improve the scientific workflow and ensure the data is processed data published more efficiently.

To mitigate issues with the high entropy mantissa bits, researchers use lossy compression. Lossy compression is capable of achieving higher compression ratios by introducing controlled error into the data [9, 20, 21] and has also been shown to increase I/O bandwidth [8]. However, these algorithms introduce a uniform amount of reduction across the entire data, which is not always the result researchers desire. In many cases, data has areas of interest that researchers wish to keep with higher fidelity. Data sampling is another data reduction method that achieves higher levels of reduction by saving a sparse subset of the data and discarding the rest. By using data sampling, researchers are able to preserve areas of their data with full precision, while less important background data has a higher tolerance to distortion. While leading data sampling approaches have been shown to retain higher data quality within regions of interest than lossy compression algorithms, these approaches are capable of being pushed further.

The research presented in this thesis contributes to the area of data sampling within scientific floating-point simulation data by integrating both spatial and temporal aspects of the input data. First, we propose the concept of spatiotemporal sampling and how it works in general. Then, we investigate the effect algorithmic configurations have on this method and apply existing concepts to yield an optimal input configuration. Lastly, we compare the effectiveness of our sampling method to existing methods.

The remainder of this thesis is organized as follows. Chapter 2 is a discussion of related work in data reduction motivation and schemes. Chapter 3 provides the background knowledge explicitly related to reduction schemes leveraged in our sampling method. Chapter 4 explains how we specifically use those schemes to form our spatiotemporal sampling method its configuration parameters. Chapter 5 presents the architecture needed and experimental setup, including input parameters and datasets. Chapter 6 presents the final results of our data reduction scheme in terms of throughput and post-reconstruction quality. Finally, we give our conclusions and suggestions for future work in Chapter 7.

Chapter 2

Related Work

As complex HPC applications continue to generate even larger datasets, the I/O bottleneck found on HPC systems is complicated further. Researchers often leverage various forms of data reduction to alleviate this issue, including data compression, time-step selection, and data sampling.

2.1 Data Compression

Data compression algorithms are a data reduction technique that encode the input data to represent it using fewer bits. Such algorithms are divided into two categories: lossless and lossy compression. Lossless algorithms compress data using Huffman or arithmetic coding techniques while ensuring they can decompress the data without introducing any error. However, when encoding scientific floating-point data, these algorithms suffer from limited compression ratios of up to 2×. This limitation is due to the high entropy found in the binary representation of floating-point data [31]. This is unacceptable for scientific simulations like the Community Earth Simulation Model (CESM) that need a reduction of 10× or higher [4].

Conversely, lossy compression algorithms are capable of achieving much higher compression ratios by approximating parts of the data. Leading lossy compression algorithms, SZ [9, 32, 20] and ZFP [21], operate through data transformations, truncations, curve fitting models, or some combination of these techniques, allowing them to manage the high entropy found in scientific floating-point data. While this approach achieves higher compression ratios and throughputs, it does so at the cost of data fidelity. With this in mind, these algorithms control the error they

introduce through a user-specified error bound, ensuring they only introduce an acceptable amount of data distortion. Overall, these algorithms are capable of achieving high compression ratios while introducing a limited amount of error [33].

Be that as it may, lossy compression is not always an ideal solution. In many cases, domain scientists consider certain regions more valuable than others, such as the eye of hurricane simulations [19] or the dark matter halos in cosmological simulations [18]. While researchers would want to save these regions with full precision, to the best of our knowledge, no generic lossy compression algorithm exists that applies different levels of compression to different regions of the dataset. Furthermore, many scientific simulations change slowly over time, yet current leading lossy compression algorithms, SZ [9, 32, 20] and ZFP [21], do not leverage the temporal aspect of data at all.

2.2 Time-Step Selection

Another popular data reduction technique researchers use is time-step selection. Most scientific simulations have some temporal property that evolves over increments of time (time-steps) throughout the lifetime of the simulation. During these time-steps, the simulation records necessary information and checkpoints in case the simulation needs to be restarted. However, as simulations are becoming even more complex, it is becoming less tractable to record all time-steps of a simulation. For instance, the Hardware/Hybrid Accelerated Cosmology Code (HACC) [13] consists of 625 time-steps corresponding to moments in time from 5 million years back to the current day. Each time-step consists of 524,288 particles, yielding up to 220 TB of data per snapshot and 22 PB over the entire simulation [26, 35]. The sheer size of time-series simulation data makes post-hoc visualization and analysis of multiple time-steps overwhelming.

To alleviate this issue, researchers save only a select subset of time-steps that are representative of the simulation as a whole and can be used to reconstruct the time-steps that they did not save. For example, assume we choose to store time-step t_k. Following this, researchers then need to decide if the next chronological time-step, t_{k+1}, is unique enough to be stored as well or is similar enough to be estimated by t_k. After comparing the two, if they are similar enough, researchers do not need to use storage to save t_{k+1}, as t_k is a sufficient representation. Overall, time-step selection is a process that reduces the number of time-steps the user has to save, reconstruct, visualize and analyze.

Currently, there are many different approaches to time-step selection. First, the most basic form of time-step selection is to remove time-steps periodically then reconstruct this data using interpolation post-hoc. While this approach has a low overhead, it suffers from low quality as time-varying datasets change in complex patterns and at unknown frequencies. Next, another approach to time-step selection is to utilize user input to manually specify areas of interest to visualize smaller portions of the time series [30, 22, 3, 2] or by visualizing the hierarchical state transition relationships [12]. However, these visual inspection processes can become labor-intensive and relatively slow. Thus, for today's large-scale HPC datasets, automatic selection techniques are more preferred. Such techniques include choosing the most representative time-steps to save based on time-variant features [2], utilizing importance curves [37] or utilizing information theory [39].

While these various time-step selection approaches work well with slow uniformly changing data, there is a significant loss of information when they are used with data that changes more sporadically over time. By only saving a subset of time-steps, any abrupt changes in the simulation could be missed. Therefore, to ensure important events are not lost, we must preserve some information from each time-step.

2.3 Data Sampling

One last data reduction technique researchers use is data sampling. Data sampling algorithms use statistics to select a representative subset of data values to store rather than all the data. This process enables data scientists and analysts to study, transport, and store a smaller quantity of data more quickly. However, when sampling data, it is essential to consider the sample size being taken in relation to the error that sample size introduces. While a larger sample size may be unnecessary and include redundant information, too small of a sample size may lose too much information.

Currently, various existing sampling methods exist, with each one having different strengths and weaknesses. First, systematic sampling creates a sample by extracting a data value at a specific repeated interval [23]. Next, simple random sampling randomly selects a subset such that it is an unbiased representation of the dataset. Every item in the dataset has an equal probability of being selected in the sample; thus, no bias is introduced. It is a fundamental method that researchers often use as a component of more complicated algorithms. No prior knowledge of the data is needed

before one uses random sampling, and there is no restriction on sample size. Likewise, stratified sampling divides data into subsets, then randomly collects samples from each group [38, 36].

While the previous sampling methods maintain the original data distribution, mean, and other statistical properties, they do not consider a data point's value nor importance to the user. As we mentioned previously, some datasets consist of features that researchers wish to preserve at a higher level than other parts of the dataset. In this case, users will want to bias samples in these areas. Biasing samples based on the importance of their value or location enhances the overall visualization and analysis process by ensuring the preservation of regions of interest.

Importance-based sampling operates on the notion that certain data points are inherently more important to the user based on value, location, or other metrics. Nouanesengsy et al. developed a basic adaptive sampling approach that uses a user-defined importance function to determine the region of interest [27]. However, for generic sampling algorithms, this importance function should be constructed without prior knowledge of the dataset. Biswas et al. operate under the assumption that rare data values are more valuable to the user and should be sampled at a higher rate than other data points [6]. Specifically, their method constructs a histogram of the dataset, then assigns each histogram bin an importance factor such that it gives fuller bins a lower priority and emptier bins a higher priority. Their method uses this importance factor when determining whether or not to keep the data point in the sample set, giving a bias to more rare values.

While previous works show importance-based sampling methods achieve higher levels of quality than compression in regions of interest, they still lack in critical areas. Specifically, these methods do not fully utilize their sampling budget by disregarding any temporal aspects within the data.

2.4 Summary of Our Contributions

Currently, there are many different data reduction methods researchers are able to choose from. However, each of them lacks in critical areas that limit their capabilities. First, while current industry-standard generic lossy compression algorithms and importance-based sampling methods are both powerful, neither leverage any available temporal aspects of the data they are reducing. Likewise, while time-step selection approaches do leverage the temporal aspect of simulation data, they are prone to missing critical turning points in the simulation as they keep only a subset of

time-steps. This illustrates the gap between spatial and temporal data reduction.

The research we detail in this thesis aims to bridge this gap by combining the two ideas into a single sampling process. Specifically, our work integrates both the spatial and temporal aspects of the input data to improve the usage of the available sampling budget. Using this approach, we demonstrate how leveraging spatial and temporal redundancies within the data results in higher post-reconstruction quality at the same reduction rate as other sampling algorithms.

Chapter 3

Background

3.1 Simple Random Sampling

With all forms of data sampling, a sampling ratio, also known as sampling rate, is the explicit parameter to specify the number of samples taken from the original data. This parameter is driven mainly by the user's storage constraint. It is calculated as the ratio between the desired output size compared to the total original data size per time-step. Simple random sampling gives each data point an equal probability of being included in the sample, limited by the sample ratio. This method works well with data of unknown distributions as all samples are unbiased and maintain statistical quantities such as mean and standard deviation. For each element in the dataset, a random number ξ is generated and compared to the user-specified sample ratio α, where $\xi, \alpha \in [0, 1]$. If $\xi < \alpha$, the point is included in the data sample.

Listing 3.1: Simple Random Sampling Algorithm

```
INPUT: n: original data set to be sampled
INPUT: α: user-specified sampling percentage
OUTPUT: samples: array of sample values

    // Take samples randomly
    for i in range(0, sizeof(n)):
        float ξ = random(); // between 0 and 1
        if ( ξ < α): samples.push_back(n[i]);
```

3.2 Importance-Based Sampling

Importance-based sampling assumes that rare data values are more important to the user. Thus, a bias is given to data points of rare values, while more frequent values are less likely to be a part of the sample. Biswas et al. present an importance-based sampling method that assigns an importance factor to each data point such that the resulting samples over-represents the rare data values without completely ignoring more common values [6]. The importance factor is calculated using a histogram of the data values within a single time-step such that data points that fall into less full bins have a higher probability of being included in the sample. Once the importance factors are calculated, a random number ξ is generated and compared to the importance factor $IF(i)$ per data point i, where $\xi, IF(x) \in [0, 1]$. The point is included in the data sample if $\xi < IF(i)$. The resulting sample set meets the user-specified sample ratio while maintaining high data fidelity in more important regions.

Listing 3.2: Importance-Based Sampling Algorithm

```
INPUT: n: original data set to be sampled
INPUT: α: user-specified sampling percentage
OUTPUT: IF: Importance Factor per bin
OUTPUT: samples: array of sample values

    max_per_bin = α / bins; // Calculate maximum number of samples per histogram bin
    frequencies = sort(build_histogram(n)); // Build and Sort Histogram of Values

    while(i < bins): // Calculate Importance Factor per Bin
        if (frequencies[i] < max_per_bin): // Keep all of these items
            IF[i] = frequencies[i];
            i++;
        else: // Can't keep all items in bin
            for j in range(0, bins):
                IF[j] = max_per_bin;
            break;

    // Normalize Importance Factors
    for i in range(0, bins):
        IF[i] = IF[i] / frequencies[i];

    // Take samples randomly
    for i in range(0, sizeof(n)):
        float ξ = random(); // between 0 and 1
        if ( ξ < IF[i]): samples.push_back(n[i]);
```

3.3 Temporal Selection

Time-step selection is the process of analyzing the difference between sequential, chronological time-steps to determine which steps provide a representative overview of the entire data series. For example, assuming the previous time-step (t_{k-1}) was previously selected, we need to decide if the current time-step (t_k) is different enough to justify the cost of storing it in addition. If t_k, is similar enough to t_{k-1}, we do not need to select it as t_{k-1} is a sufficient representation.

3.4 Data Reconstruction

There are various existing methods for reconstructing the data back to full resolution based on a collection of samples. Interpolation is a process of approximating the value of a non-given data point given the values of a discrete set of the known surrounding data points. In our workflow, we use our group of samples to interpolate the intermediate values not included in the sample, then compare this full-resolution data with the original dataset to see how well a sample we saved. One of the most common and simplest interpolation methods is a piecewise constant interpolation, also known as nearest-neighbor interpolation. This method assigns the non-given data the value of the nearest sampled point. This process can be fast as it does not consider any of the other neighboring points, but this causes the resulting data to have blocky artifacts.

Linear interpolation is a method that uses the curve fitting of linear polynomials to estimate the value of non-given data using the surrounding samples. This method usually produces a higher quality reconstructed data than nearest neighbors, but at a computational cost. Using a Delaunay triangulation with linear interpolation is a fast method that divides the domain into triangles, defined by the three vertices chosen from the sample, that form a plane surface [10]. We use a linear interpolation-based reconstruction for our experiments using a Delaunay triangulation reconstruct, as it is a balanced trade-off between quality and speed. In general, the higher-order the interpolation, the better the quality but the slower the process.

3.5 Data Quality

After reconstructing a time-step from the gathered data samples, we assess the quality of the resulting data. Assessing quality based on the visual representation can be too subjective. Thus, we

use the peak signal-to-noise ratio (PSNR) to represent the quality between the reconstructed data and the original data. PSNR is the ratio between the original and reconstructed data, measured in decibels (dB). The higher the PSNR, the better quality the reconstructed data is. PSNR is based on the mean-square error (MSE), the cumulative squared error between the original and reconstructed data values. We describe MSE as Equation 3.1, where n is the number of data points in the dataset, Y_i is the original value and $\hat{Y_i}$ is the reconstructed value. Equation 3.2 details PSNR, where max_val and min_val are the maximum and minimum of the original dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 \tag{3.1}$$

$$PSNR = 20 * log10((max\_val - min\_val)/\sqrt{MSE}) \tag{3.2}$$

Chapter 4

Spatiotemporal Sampling

4.1 Overall Concept

Within most HPC scientific simulations, larger areas transform slowly over time, with smaller regions changing more drastically, such that there is a visual difference between the two. For instance, Figure 4.1 visualizes three time-steps of a simulation of the 2003 Hurricane Isabel, specifically the pressure variable [15]. We divide each time-step into smaller sub-regions. In this dataset, each time-step's defined region of interest is the hurricane eye, as weather forecasters and meteorologists study it to determine when the hurricane is gaining strength and when it is necessary to take precautions.

Our goal per time-step is to preserve data fidelity in the region of interest as much as possible, at the cost of some precision in other areas of the dataset. To do so, we use the importance-based method by Biswas et al. [6] (see Chapter 3.2) as it has been shown to yield higher accuracy in regions of interest than other methods.

Over time, some regions of the dataset, like region A, change drastically as time progresses. In general, this area of frequent change corresponds to the region of interest. As such, we want to save this region with higher accuracy. However, the unchanging regions, like region C, are background information and therefore are most likely not as important to the domain scientists. The unchanging regions are less important than the areas of frequent change, so we tolerate more error in them. Lastly, regions like B are some mix in between, where there is little change over time. Thus, we need to keep accuracy in those regions, but it is not as important as A.

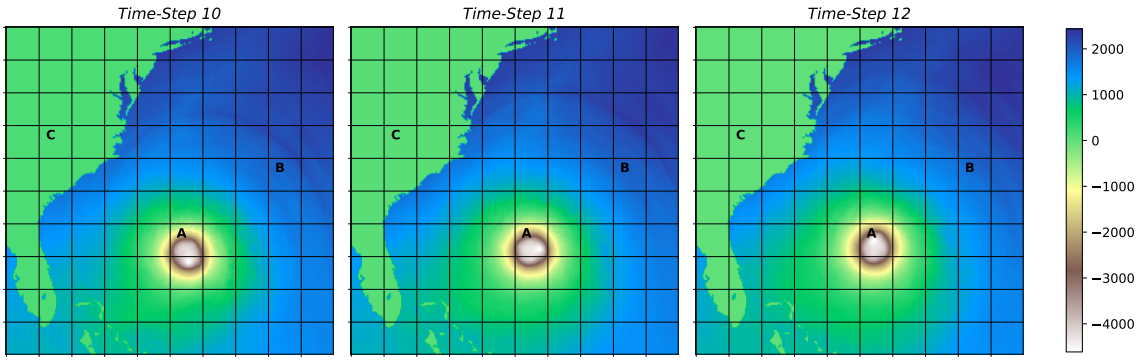


Figure 4.1: Hurricane Isabel divided into regions

For regions like C that are consistent over time, the idea of temporal selection can be applied in this region, as the data in region C in time-step 11 is a sufficient representation of the data in the corresponding region in time-step 12. However, time-step selection would not be a good fit for region A since data changes quickly over time; we want distinct information from every time-step in this region to ensure we save a sufficient amount of information.

Such is the motivation for our spatiotemporal sampling method. Not only is it essential to select samples based on their relative location and value within one time-step, but also to acknowledge and leverage redundancies over time. Doing so enables our sampling method to provide higher post-reconstruction quality than other current data sampling algorithms.

Given a data series with multiple time-steps divided into a certain number of sub-regions, we compare each corresponding region to its chronological neighbor. From that comparison, we determine if it is different enough to be incorporated into the temporal selection or similar enough to be represented by the previously selected data. To accomplish this, we first need to quantify what it means for two regions to be similar or different enough.

4.1.1 Histogram-Based Reuse

The first way we determine if two regions are similar is to compare the distribution of data values in each. Histograms are lightweight in terms of storage and add little computational overhead. Thus, they are a valuable way to compare two regions quickly. We first construct a histogram of the data values within each region in two chronologically neighboring time-steps, 11 and 12, from Figure 4.1. Next, we use histogram intersection (see Equation 4.1) to determine if the two distributions are similar enough to utilize the data from time-step 11 for time-step 12 as well.

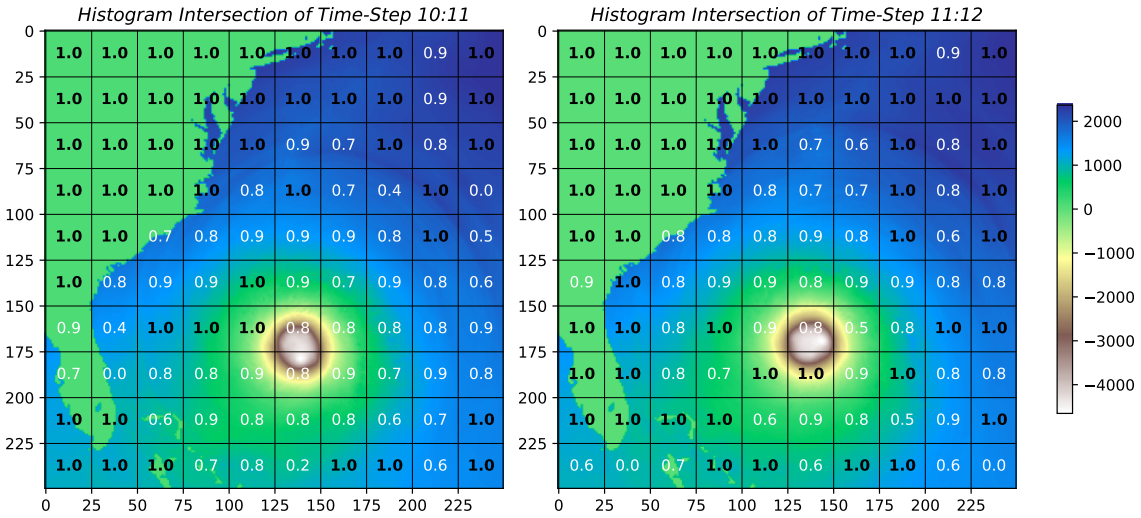


Figure 4.2: Hurricane Isabel histogram intersection

Here, p_i is the number of elements in the ith bin of the histogram of the previous time step, and q_i is the number of elements in the corresponding ith bin of the current time step histogram, and n is the total number of bins. We normalize the intersection by dividing by q_i, such that the results are always between 0 (no intersection) and 1 (identical distributions).

$$\frac{\sum_{i=1}^{n} min(p_i, q_i)}{\sum_{j=1}^{n}(q_i)} \tag{4.1}$$

We only choose to reuse previous information in a region for our implementation if 100% of the previous and current histograms for that region intersect. This means that the histograms of the two regions are identical; thus the data distribution of that region has not changed between the two time-steps. Figure 4.2 shows the amount of intersection per region between time-steps 11 and 12. The areas marked with 1.0 mean the histograms are 100% intersected. Thus, the data in those regions will not take new samples from time-step 12 and instead refer to the data in time-step 11.

4.1.2 Error-Based Reuse

Our second method for determining similarity uses the error between the two temporally corresponding regions. This method allows the user to set an error tolerance based on each region's root mean square error of each region, as described in Equation 4.2, where p_i is a value in the previous time-step and q_i is the corresponding value in the current time-step. When the root mean squared

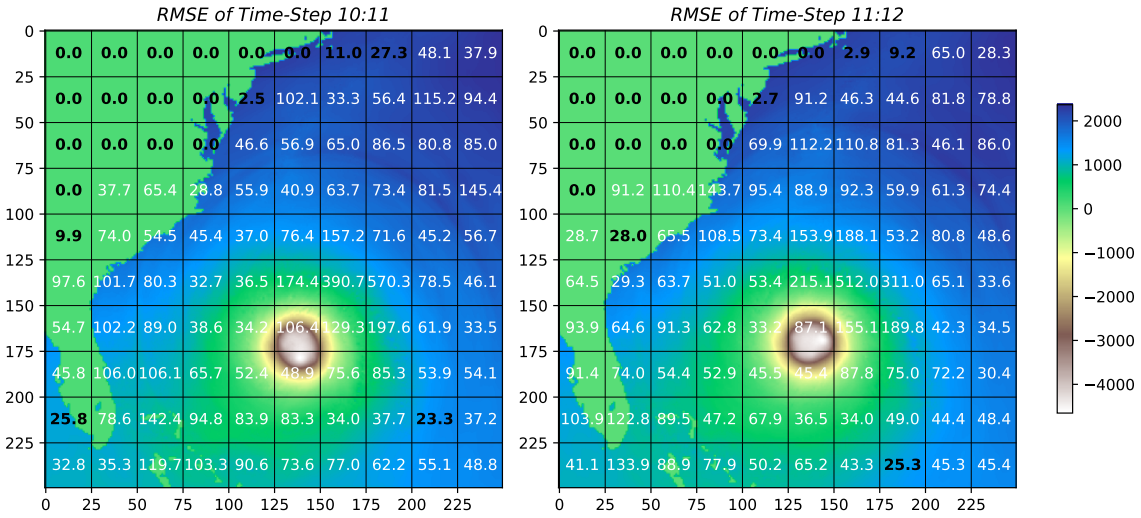


Figure 4.3: Hurricane Isabel root mean square error

error is less than the specified tolerance, regions are considered similar enough to be represented by previous information. If the error of a region is greater than the threshold, it cannot be represented by the previous time-step as it would introduce more error than the user wants to allow.

$$\sqrt{\frac{\sum_{i=1}^{n}(p_i - q_i)^2}{n}} \tag{4.2}$$

For example, Figure 4.3 shows the error per region between time-steps 11 and 12. If the user threshold was 28.0, then regions less than that tolerance would not be saved in the sample, as the information in 11 is representative of that in 12 for that region.

4.1.3 Similarity Metric Comparison

We leave the selection of similarity metrics to the user, as each has advantages and disadvantages. Thus, the best metric heavily relies on the dataset and user constraints. First, we analyze the speed of the two methods. It is relatively fast to build and calculate the intersection of histograms. Therefore, two time-steps of Hurricane Isabel with 200 regions can be compared in 14 MB/s. The calculation of error takes longer, running in 13 MB/s for the same data. The difference in speed between the two processes grows as the number of regions increases.

Second, we analyze the amount of storage it takes to calculate similarity. Histograms are a lightweight solution when using a relatively small number of bins. To compare each region of the

current time-step with the previous one, we only need to have access to the region histograms. Thus the amount of storage needed in bytes is calculated as Equation 4.3, based on the number of data values per time-step (n), the number of regions (r), and the number of bins (b).

$$n + (r * b * size(int)) \tag{4.3}$$

When using the error-based method, we need to have access to all samples taken from the previous time-step to calculate the error. Thus, as the sample ratio increases, so does the number of samples taken and the number of computations needed to calculate the root mean squared error. The amount of storage needed to calculate similarity with the error-based metric is calculated as Equation 4.4, based on the number of data values per time-step (n) and the sample ratio (s). Thus, depending on the sample ratio, the error-based method may introduce more storage overhead than the histogram-based method.

$$n + (n * s) \tag{4.4}$$

Lastly, we address the quality of each method. While histograms are a lightweight and fast solution, they lack spatial awareness. The distribution of the values within a region might remain the same but might have changed in location within the region. Thus if we deem two regions similar based only on identical value distributions, we may introduce too much error, yielding a lower quality. The error-based method for quantifying similarity is based on the user's specified error tolerance, making it easier to bound the amount of error and, in general, yielding a higher quality post-reconstruction.

The main attribute to the difference in quality can be seen in the differences between Figures 4.2 and 4.3 where we analyze which and how many regions are considered similar per method. When using histograms to determine similarity, we find 24% of regions eligible for reuse between the 10th and 11th and 22% between the 11th and 12th. However, the error-based method is more rigorous, as it only reuses 10% to 14% of regions from time-steps 10 to 11 and 11 to 12, respectively. The amount of storage required to store the samples of time-step 11 is shown in Figure 4.4, where the error-based method reused fewer regions; thus, it takes more storage to save more samples than the histogram-based method. Since this finer similarity metric reuses fewer regions, as it is bound to the user-specified error tolerance, it generally yields higher quality. For example, we analyze region

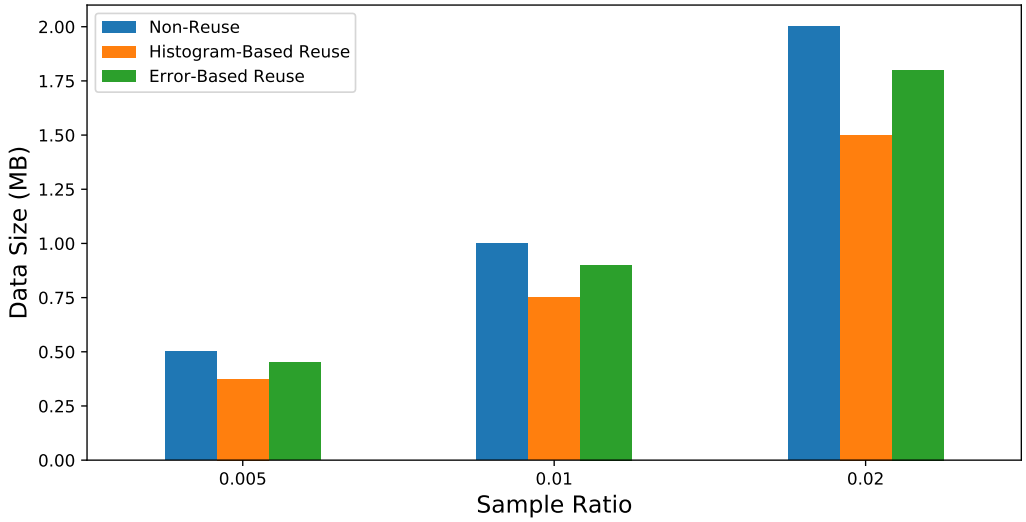


Figure 4.4: Hurricane Isabel reduced data sizes

B of Figure 4.1 as it was labeled a region of medium levels of change over time. The histogram-based method determined to reuse this region, as the histograms between time-steps 10 and 11 were identical. However, the error-based method found that those two regions had a root mean square error of 45.2, greater than the specified error tolerance of 28.0. Thus, if the histogram-based method were used, it would allow in more error than the user wanted even though it is faster.

Regardless of similarity metric, some amount of regions are specified for reuse in each time-step. Since data sampling works by saving a specified amount of samples, as given by sample ratio, we want to fill the sampling budget as best as possible. The non-reuse method will fill the sample budget nearly completely. However, since the histogram-based and error-based reuse methods reuse some data from the previous time-step instead of taking new samples from the current time-step in some regions, they do not use the entire budget. This trend is seen in Figure 4.4, where both reuse methods only fill about 70% of the allocated budget. Since our method is centered around the notion that the user sets the sample ratio to meet the storage allocated, we aim to meet this space as near as possible. Thus, we add a layer of random samples to the sample set until the allocated space is filled.

4.2 Configuring Parameters

Our sampling scheme has several input parameters designed to give the user the most flexibility with our algorithm. However, this freedom may become a burden to the user if specific details about the data are unknown. For example, we need to know the number of bins to use, error threshold, and sub-region dimensions before the sampling process. If the user randomly selects a less than optimal value for one of these parameters, the data quality can significantly be lower than expected. Thus, we provide the user with a dynamic parameter assistance tool to aid in the parameter setting process.

4.2.1 Determining Number of Bins

Our sampling method gives a sampling bias to data points whose values are rarer. We determine the rarity of each value based on the sorted histogram of each time-step. We also leverage smaller histograms per region when using the histogram-based method to determine when to reuse previous information. This heavy reliance on histogramming means that we need to determine an optimal number of histogram bins early on in the process to ensure consistency throughout the data reduction process. Having a consistent number of bins is critical for the histogram-based reuse method, as if we are using histogram intersection to determine reuse, we need to ensure that the bin ranges are the same such that we have a fair comparison.

When selecting the number of bins for our histograms, we could use a random arbitrary number or estimation. However, we could not ensure that this yields an optimal sample. When using the sampling method provided by Biswas et al. to gather samples, the number of bins is important to set wisely, as, with too few bins, rare important values may be placed into the same bin as less important values. Likewise, the histogram will be too finely distributed with too many bins, and the overall algorithm will be slowed down. Moreover, the number of bins affects the amount of histogram intersection with the histogram-based reuse method. In general, with fewer histogram bins, it is easier to have each bin intersect as there are fewer ways to distribute the data. With more bins, the data is more distinctly identified and categorized; thus, it is more challenging to have an identical histogram as it is less tolerable to minor variations in the data values. This trend is shown in Figure 4.5, where we analyze the amount of histogram intersection in regions A, B, and C between time-steps 11 and 12 of Figure 4.1. In general, as the number of bins increases, the amount

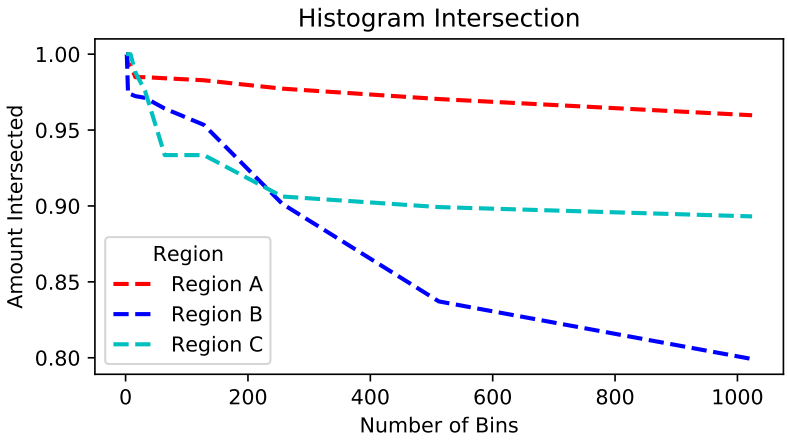


Figure 4.5: Hurricane Isabel distribution intersection

of intersection decreases. This trend is more dramatic in regions of entropy, like region A. This trend is essential to understand and leverage as, with too many bins and too few intersections, our histogram-based reuse method is left un-optimized and rarely utilizes previous data. On the other hand, fewer bins yield more intersection, which will hyper-inflate the number of regions determined similar, resulting in excess error.

To assist users in determining an optimal number of histogram bins, we run a pre-processing step using existing algorithms and the first time-step. To set a consistent bin width, we need to know the entire possible range of the data values. Since we aim to set the bin ranges once and use them consistently, we must ensure that we account for all possible values. We use this range as the minimum and maximum edges in the histogram. Next, we need to determine the bin width or the total number of bins to yield an optimal distribution. Sturges' rule was the original proposed method for estimating an optimal number of bins and is widely recommended. It calculates the number of bins (K) as Equation 4.5, where n is the number of data points to histogram. While this algorithm is generally the default, it tends to over smooth the histogram with large datasets and is only optimal for gaussian data. Therefore, we use Doane's rule, a modification to Sturges' that works better with non-normal datasets [16]. Doane attempts to account for the skew of the data. The algorithm for this is shown in Listing 4.1.

$$K = 1 + log_2(n) \tag{4.5}$$

Listing 4.1: Doane's Rule Algorithm

```
INPUT: n: original data set to be sampled
INPUT: data_range: range of data values over the entire series
INPUT: mean: mean of data values over the entire series
INPUT: stdev: standard deviation of data values over the entire series
OUTPUT: numBins: number of bins recommended

    sg1 = sqrt(6.0 * (n.size() - 2) / ((n.size() + 1.0) * (n.size() + 3)));

    if (stdev > 0.0):
        temp = n - mean;
        temp = temp / stdev;
        temp = pow(temp, 3);
        sum_temp = std::accumulate(temp.begin(), temp.end(), 0.0);
        mean_temp = sum_temp / temp.size();
        binWidth = data_range / (1.0 + log2(n.size()) + log2(1.0 + abs(mean_temp) / sg1));
        numBins = data_range / binWidth;
        return numBins;
```

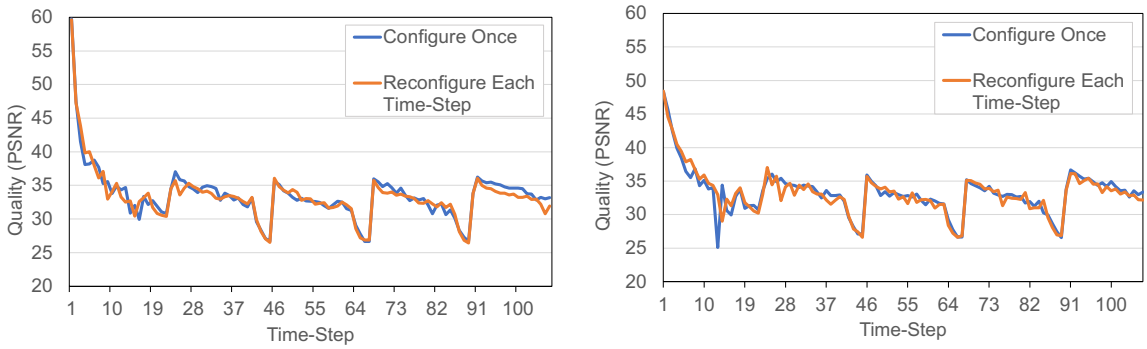
Secondly, we use Scott's rule, as it is more statistically rooted by taking data size and variability into account. It also works well with large datasets, which is when our data reduction scheme is needed most. In this method, the bin width is proportional to the standard deviation of the data, which is not very robust to outliers. The algorithm for this is shown in Listing 4.2.

Listing 4.2: Scott's Rule Algorithm

```
INPUT: n: original data set to be sampled
INPUT: data_range: range of data values over the entire series
INPUT: mean: mean of data values over the entire series
INPUT: stdev: standard deviation of data values over the entire series
OUTPUT: numBins: number of bins recommended

    binWidth = pow((24.0 * sqrt(pi) /sizeof(n)),(1.0 / 3.0)) * stdev;
    numBins = data_range / binWidth;
    return numBins;
```

In our pre-processing step, we use both Doane's and Scott's rules to provide the user with a range of bins to consider. Instead of simply presenting these recommendations to the user, we also collect a small number of samples with these bins and reconstruct the data to estimate the resulting data quality. Based on this estimation, our algorithm selects the number of bins corresponding to the highest quality. Thus, the overall overhead of this pre-process depends on the time it takes to sample, reconstruct, and analyze the data twice. Since we only need an estimation of the quality to assess



(a) Non-Reuse.

(b) Error-Reuse.

Figure 4.6: Consistent vs. varying number of bins

which number of bins is better, we use an OpenMP accelerated CPU version of nearest neighbors reconstruction. This method is faster but yields lower quality than other reconstruction methods. This is an acceptable trade-off, as we only need to know the quality trend and not necessarily the exact expected quality.

When using the histogram-based reuse method, it is crucial to maintain the same bin width, to ensure a fair comparison. However, when using the non-reuse or error-reuse method, we could update the number of bins for each time-step. Our experiments in Figure 4.6 find no improvement in configuring the number of bins once at the beginning or re-configuring at each time-step. This is true for our experiments, as our sampling algorithm is designed to work with datasets that progress smoothly over time. Thus, using the number of bins optimal for the first time-step yields the same quality as if we were to recompute the optimal number of bins for each time-step independently.

4.2.2 Determining Error Threshold

When using the error-based reuse method, we use the root mean square error to determine if two temporally corresponding regions are similar. Regions are considered similar if their error is less than the user-specified error threshold, designed to limit the amount of error allowed when reusing data. However, the user may not know what threshold would yield the best result. With a threshold too loose, too much data from the previous time-step would be reused, even if it is not a sufficient representation of the updated values of the current time-step. However, very little data will be reused if the threshold is too strict, leaving our algorithm un-optimized. It will add an extra temporal overhead of checking for similarity but finding nothing similar enough to leverage, yielding

the same quality as the non-reuse methods.

As a generic standard to determining the error threshold, we calculate the difference to each corresponding data value between the first two time-steps of the series. After calculating the difference between these steps, we sort the differences from least to greatest. Lastly, we calculate the third quartile of this list and present this as our proposed error threshold. This process is detailed in Listing 4.3.

We have found that using the third quartile of the sorted list of errors is the best generic error threshold, as it generally yields one of the highest quality, post-reconstruction. This method works well, as the user does not need to know any prior information on their dataset.

Listing 4.3: Generic Error Threshold Algorithm

```
INPUT: n0: time-step 0 of original dataset
INPUT: n1: time-step 1 of original dataset
OUTPUT: error_threshold: error threshold recommended

    vector<float> error(sizeof(n));
    for i in range(0, sizeof(n0)):
        error[i] = abs(n0[i] - n1[i]);
    sort(error.begin(), error.end());
    error_threshold = error[int(sizeof(n0)*0.75)]; // 75th percentile AKA 3rd quartile
```

4.2.3 Determining Region Dimensions

It is critical to set an appropriate region size to yield an optimal performance of our spatiotemporal sampling method, as a region size too small or too large affects both overall quality and throughput. For example, Figure 4.7 shows the hurricane dataset divided into differently sized regions. If we used a region size too large, it would be more difficult to consider two temporally corresponding regions similar, as there is more data inside.

The more regions we divide the dataset into, the lower the amount of data within them, making the data values very localized. This raises the probability that this small amount of data points has not changed over time, potentially yielding more reuse and generally a higher quality sample. However, the more regions, the more similarity comparisons are needed, which slows down the overall process. Thus, we need some balance between a small and large region size.

To assist the user in determining an optimal number of regions for their dataset and constraints, we run a pre-processing step to estimate the throughput and amount of regions reused per

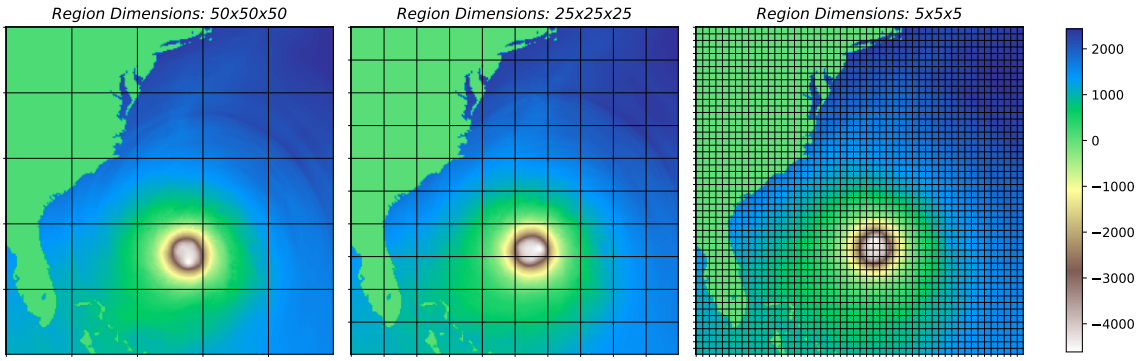


Figure 4.7: Hurricane Isabel regions

number of regions. We provide a line plot to the user to visualize the trade-off between the two metrics. In general, as the number of regions increases, the quality increases, but the throughput decreases. Thus, the optimal number of regions depends on what is more important to the situation: quality or throughput. Given this trade-off plot, one can choose the number of regions that are best for their constraints. While an exhaustive study of testing every possible region dimension would yield a more specific plot, doing so would introduce a costly overhead that outweighs the pre-processing benefits. Thus, we only test a distributed subset of 10 region sizes, as it adequately represents the overall trend while adding little overhead.

4.3 Sampling Process

In this section, we detail the specific steps that comprise our spatiotemporal sampling algorithm. This sequence of processes runs for all time-steps in the series. Note that steps 1, 2, 3, 6, and 7 are leveraged from the importance-based method of Biswas et al. [6]; we refer to this process as the non-reuse method. We use the non-reuse method for the first time-step as there is no previous data to utilize.

4.3.1 Step 1: Data Histogram Creation

Regardless of reuse, the first step is to construct a histogram of all the data values in the current time-step. The bin range of this histogram is based on the maximum and minimum values of the current time-step.

4.3.2 Step 2: Data Histogram Sorting

The next step is to sort the histogram bins from least to greatest. This resulting list of bins is then used to develop the acceptance function that biases the values that fall into bins with the lowest amount.

4.3.3 Step 3: Acceptance Function Development

Using the sorted histogram, we develop an acceptance function that sets the acceptance rate for each bin. If a data point's value falls into that bin range, it is compared to the corresponding bin acceptance rate. Using the user-specified sample ratio, we determine the target number of samples per bin, which, ideally, is evenly distributed. Thus the maximum target number of samples per bin equals the sample ratio divided by the number of bins. We then iterate over the sorted histogram to determine if the number of points in the bin is more than the target maximum. If so, we update the bin value to equal the target maximum. If there are fewer data points than the maximum, we do not update. If a bin does not reach the maximum, the unused storage is redistributed among the remaining bins. Once every bin has been processed, we divide them by the original histogram entry, creating an acceptance rate between 0.0 and 1.0. This ensures that the rare values have an acceptance probability closer to 1.0, therefore more likely to be a part of the sample, and common values have an acceptance probability closer to 0.0. This process is detailed in Listing 4.4.

Listing 4.4: Importance-Function Algorithm

```
INPUT: n: original data set to be sampled
INPUT: α : user-specified sampling percentage
INPUT: numBins: number of histogram bins
OUTPUT: IF : Importance Factor per bin

    max = α / numBins; // Calculate maximum number of samples per histogram bin

    // Build and Sort Histogram of Values
    frequencies = build_histogram(n);
    frequencies = sort(frequencies);

    // Calculate Importance Factor per Bin
    while(i < numBins):
        items_in_bin = frequencies[i];
        if (items_in_bin < max):
            // Keep all of these items
            IF[i] = items_in_bin;
            i++;
        else:
            // Can't keep all items in bin
            for j in range(i, numBins):
                IF[j] = max;
            break;

    // Normalize Importance Factors
    for i in range(0, numBins):
        IF[i] = IF[i] / frequencies[i];
```

4.3.4 Step 4: Region Histogram Construction

If we use the histogram-based reuse method, we need to construct a histogram per region of the current time-step. While these smaller histograms use the same number of bins as the full data histogram, the bin range is set to the minimum and maximum values one expects throughout the simulation's lifetime. Since these histograms are used to compare multiple time-steps, we need to ensure the bin widths stay constant for all samples. Thus, we need to use the lifetime extrema rather than for the current time-step alone.

4.3.5 Step 5: Region Comparison and Reuse

When using the histogram-based or error-based reuse method, we must compare each region to its temporally corresponding counterpart. With the histogram-based method, we calculate histogram intersection for each region with the current and previous time-steps. If they are found to be identical, they are marked for reuse. With the error-based method, we calculate the root mean square error per region. If a region has an error less than the user-specified tolerance, it is marked for reuse. If previous information is unavailable or has been reused in the previous iteration, we skip the region. This helps us to avoid a domino effect upon time for reconstruction.

4.3.6 Step 6: Random Number Generation and Sample Selection

In this step, we determine which data points will be included in the sample. First, we create a mask of zeros equal to the size of the dataset to signify which points will be a part of the sample set and which will be left out. Then, for each data point, we determine which bin the value would fall into in the global histogram and the corresponding acceptance rate for that bin. Next, each data point is assigned a random number, generated between 0.0 and 1.0. If this random number is less than the acceptance rate, we update the corresponding mask location to a 1 to signify that it is to be kept in the sample.

4.3.7 Step 7: Gathering Sample Data

Once all of the data has been processed, we need to use the previously created mask to collect all of the data locations and values for the chosen samples and append them to the final list of samples.

4.3.8 Step 8: Additional Random Sampling

Both the histogram-based and error-based methods use less storage space by reusing information in certain regions of the previous time-step instead of taking new samples for the entirety of the current time-step. Figure 4.4 shows that the non-reuse method will nearly exactly fill the user-specified sample ratio, while the reuse methods do not. Since our method is centralized around the concept of a sample ratio, we aim to collect a number of samples as close as possible. Thus, we use simple random sampling to fill the remainder of the sample budget. We only take more samples

from the regions that did not reuse this time-step to further improve the quality in those areas of significant change.

4.4 Summary

Our goal is to reduce the overall size of a data series by taking spatial samples of each time-step, and leveraging temporal similarities to reuse samples between time-steps. Figure 4.4 shows that regardless of how temporal similarities are quantified, by reusing data from previous time-steps we have the storage capacity to take additional samples to fill the user-specified sample ratio. In general, more samples yield higher reconstructed quality. Thus, we expect improvements over existing data sampling methods.

Chapter 5

Experimental Setup

With our spatiotemporal sampling method designed and implemented, we move to evaluate both our preprocessing steps and the effectiveness of our sampling method. In all of our experiments, we conduct our trials using Clemson's Palmetto Cluster [28]. Specifically, we use an R740 model 40 core Intel Xeon CPU with 372 GB of memory when running all trials. We evaluate our sampling method using three real-world High-Performance Computing datasets and three sampling ratios: 0.5%, 1%, and 2%. Table 5.1 describes the details of each data set and the input parameters we use for sampling (as described in Section 4.2).

5.0.1 ExaAM

The Exascale Additive Manufacturing Project uses exascale simulations to design Additive Manufacturing components [5, 17]. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S.Department of Energy Office of Science and the National Nuclear Security Administration. It consists of 108 time-steps of spatial resolution $20 \times 200 \times 50$. We experiment with this data set primarily to show the difference in results when using a smaller data set that has more time-steps. Figure 5.1 shows time-step 64, with the highlighted region of interest, the hottest portion of the visual.

5.0.2 Isabel

The Hurricane Isabel Data models the 2003 hurricane in the western Atlantic region [15]. This data was produced by the Weather Research and Forecast model, courtesy of NCAR, and the U.S. National Science Foundation. For our experiments, we sample the pressure variable, as it provides a distinct representation of the region of interest for this data, the hurricane eye, as seen in Figure 5.2. This dataset consists of 48 time-steps of resolution $500 \times 500 \times 100$.

5.0.3 Impact

The Deep Water Impact Ensemble dataset [29] is the collection of simulations run at Los Alamos National Laboratory to study Asteroid Generated Tsunami. These simulations help study the impact of an asteroid in deep ocean water to learn the limit of dangerous asteroids. We sample the water volume variable, V02, as it clearly visualizes the resulting water splash, the region of interest of this dataset. The data values range from 0.0 to 1.0, where 1.0 is pure water and <1.0 is asteroid debris mixed with water. For our experiments, we use 100 time-steps of resolution $300 \times 300 \times 300$, as presented in Figure 5.3.

DATASET	VARIABLE	DIMENSIONS	DATA SIZE	STEPS	SUB-REGION	BINS	ERROR
ExaAM	-	$20 \times 200 \times 50$	0.8 MB	108	$10 \times 40 \times 10$	633	0.0
Isabel	Pressure	$500 \times 500 \times 100$	95 MB	48	$25 \times 25 \times 25$	27	28.0
Impact	V02	$300 \times 300 \times 300$	108 MB	130	$50 \times 50 \times 50$	27	0.0

Table 5.1: Datasets and configurations used in experimental evaluations.

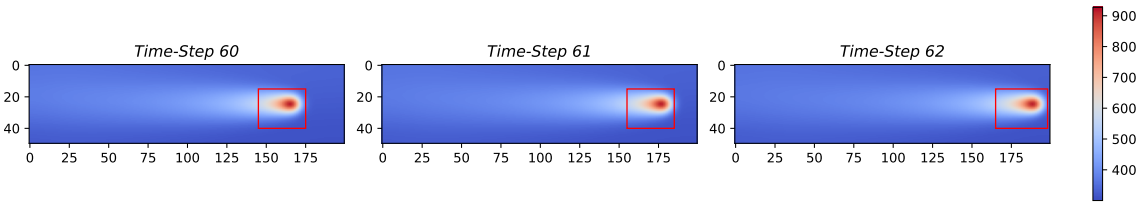


Figure 5.1: ExaAM dataset

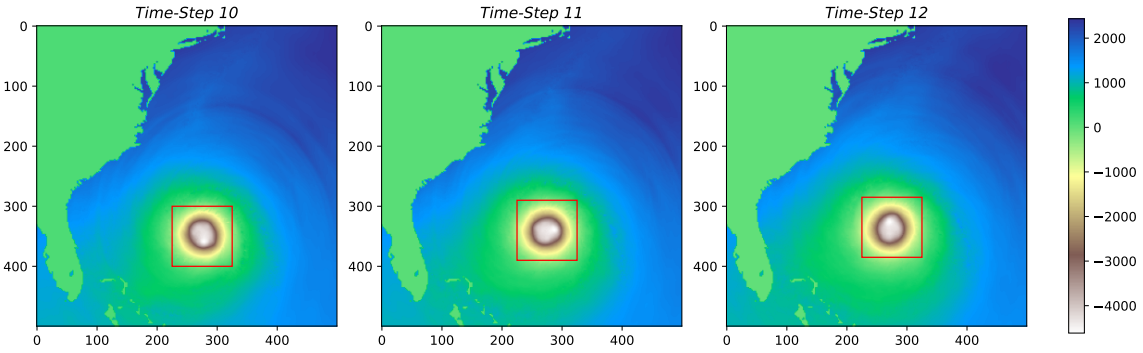


Figure 5.2: Hurricane Isabel pressure dataset

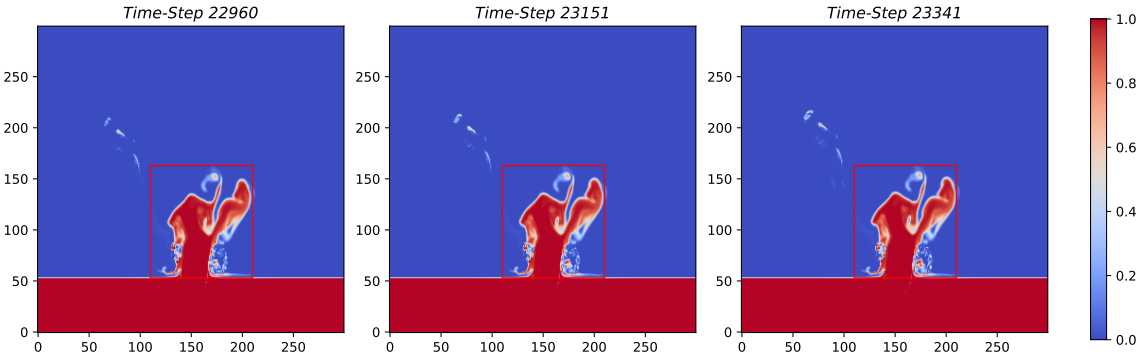


Figure 5.3: Asteroid Impact V02 dataset

Chapter 6

Evaluating Our Sampling Method

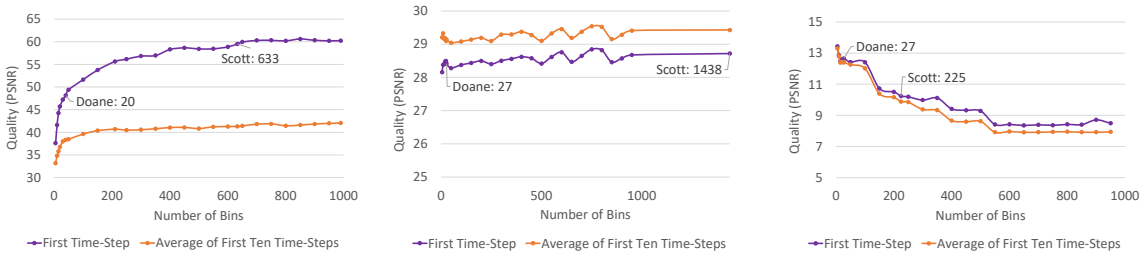
We first analyze the effectiveness of our proposed pre-processing steps to determine the number of bins, error threshold, and the number of regions for each dataset. Using those parameters, we sample each dataset with various methods and analyze the throughput of each. Next, we reconstruct from the samples and calculate the average data accuracy with the original data. Lastly, we explore changing the sampling algorithm used within our reuse method and analyze how doing so affects throughput and quality.

6.1 Configuring Parameters

6.1.1 Number of Bins

Our first pre-processing step is to determine what number of bins to use with the histograms when sampling the data. This step uses Doane's rule and Scott's rule to provide two possible optimal numbers of bins for the dataset, based on the first time-step. Using the two proposed numbers of bins, we estimate the quality of the reconstructed dataset if we were to construct with that many bins. Then we either use Doane's or Scott's proposal, based on which provided the highest quality post-reconstruction dataset. The number of bins our pre-processing step selects per dataset are listed in Table 5.1.

Figure 6.1 shows an experiment varying the number of histogram bins and the resulting quality per dataset. This shows that our pre-process's proposed number of bins yields one of the

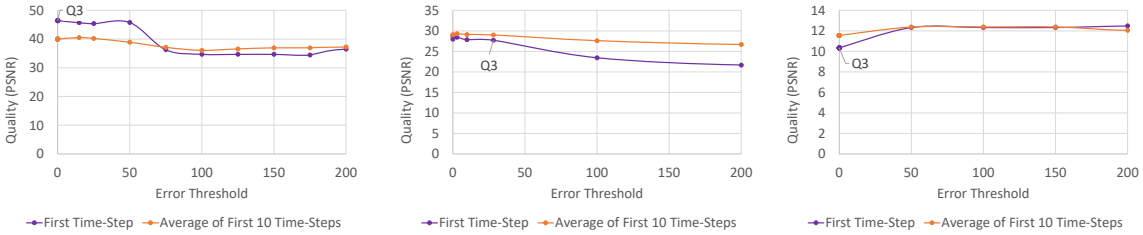


(a) ExaAM. (b) Isabel. (c) Impact.

Figure 6.1: Average quality, varying number of bins

highest qualities across datasets. Since this process determines an optimal number of bins based only on the first time-step, it is possible that it does not yield the most optimal overall time-steps. This is why we also provide the average quality of the first ten time-steps if we were to sample with that number of bins. This shows that the proposed number of bins still provides one of the highest qualities, even as the data series progresses, as our sampling algorithm is designed to utilize temporal similarities; Thus, it works best with datasets that change smoothly over time. Overall, while our method does not guarantee to propose the most optimal number of bins, it does provide a more near-optimal option than if the user were to set at random. For example, Figure 6.1a shows that with the ExaAM dataset, a user could randomly choose to use ten bins for their experiments, resulting in a PSNR of 42 dB. However, using our pre-processing step, 633 bins are used, yielding a 1.43× improvement in quality.

Even though this pre-processing step improves the overall quality of the reconstructed data, it is only as effective as its throughput. We run the data reduction process to avoid the I/O bottleneck, so the overall data sampling algorithm is useless if it itself becomes the new bottleneck. We find that with a sample ratio of 1%, this specific pre-processing step introduces an average 0.2% temporal overhead of the sampling process for the entire data series. The majority ($\geq$93%) of this process is spent in the reconstruction and quality analysis phase. However, we find this to be an acceptable temporal trade-off, as assisting the user in determining the number of bins helps yield the highest overall quality.



(a) ExaAM. (b) Isabel. (c) Impact.

Figure 6.2: Average quality, varying error threshold.

6.1.2 Error Threshold

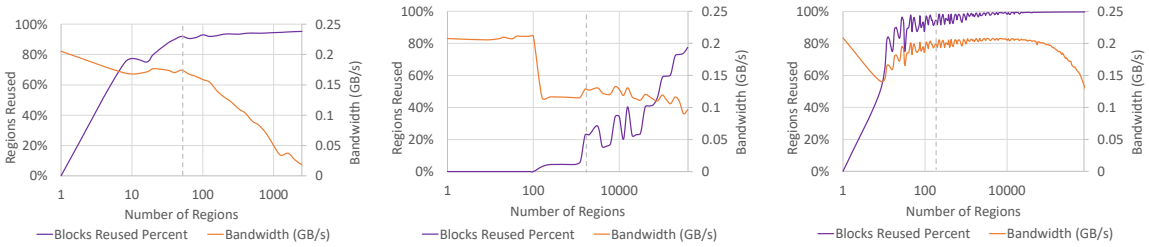
Next, we run our pre-processing step to estimate an optimal error threshold per dataset. This process calculates the sorted error distribution between the first two time-steps and takes the third quartile as the threshold. We run experiments with varying error thresholds and resulting post-reconstruction quality to prove the validity of this process. As shown in Figure 6.2, using the third quartile of error as the threshold results in one of the highest qualities across datasets. The trend holds as we continue to use it over the next ten time-steps.

We use the error threshold recommended by this pre-processing step for each dataset, as listed in Table 5.1. The ExaAM and Impact datasets both have a recommended threshold of 0.0 because they change very smoothly over time, causing the difference between the first two time-steps to be very small. Even with this stringent error threshold, there are still enough similarities between regions to consider samples reusable.

With a sample ratio of 1%, this pre-processing step introduces an average 0.03% temporal overhead to the entire sampling process, varying slightly as data size and number of time-steps varies. While the error threshold has a smaller effect on quality than the other input parameters, we still find this pre-processing step a good trade-off between the possible quality achieved and the low temporal overhead.

6.1.3 Number of Regions

Lastly, we use a pre-processing step to determine the number of regions to divide each time-step into. In this process, we compare the average percentage of regions reused from time-step t_{k-1} when gathering samples for time-step t_k, of the first ten time-steps. Our spatiotemporal sampling



(a) ExaAM. (b) Hurricane Isabel. (c) Asteroid Impact.

Figure 6.3: Evaluation of region size

algorithm utilizes more samples from t_{k-1} when we have each time-step divided into more small regions than with fewer large regions. The more regions we have, the smaller the region dimensions are, meaning the data within each region is very precise. This allows our algorithm to reuse more regions as the probability of fewer data points changing over time is lower than if we had to consider a region of more data points. Reusing more regions allows us to have access to more samples overall, which generally correlates to a higher post-reconstruction quality. Thus, if the user wants the highest qualities possible, the more regions they should specify. Consequently, having more regions means that more similarity computations have to be performed, drastically slowing down the overall sampling process. This trade-off is presented in Figure 6.3. In general, the more regions, the higher the percentage of regions reused, but the lower the throughput. We present a version of these plots to the user with their input dataset for them to choose a number of regions to focus on higher post-reconstruction quality, higher throughput, or some trade-off of both.

With a sample ratio of 1%, the average overhead introduced is $\leq$3% of the entire sampling process. Similar to the previous two pre-processes, we deem this overhead as an acceptable trade-off as it provides a more precise understanding of the quality-throughput trade-off of our algorithm. It also aids the user in choosing a configuration that will meet their standards. In the following experiments, we choose a middle-ground number, as described in Table 5.1, to have a more general amount of previous time-step utilization with an acceptable bandwidth trade-off.

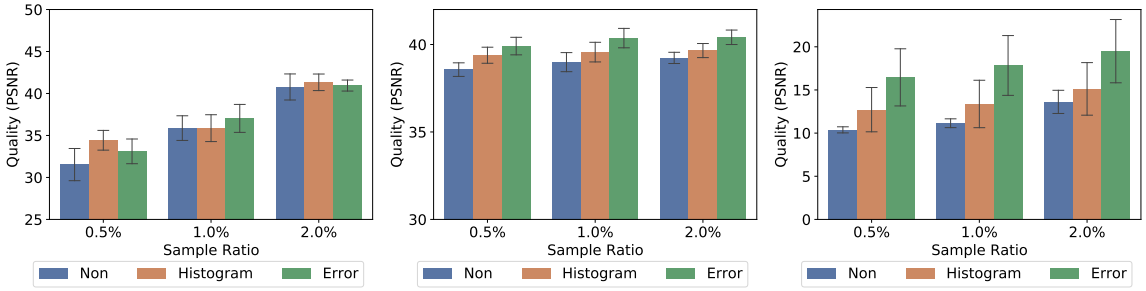
6.2 Overall Spatiotemporal Sampling

In this section, we evaluate the overall throughput and quality of our spatiotemporal sampling algorithm as compared to existing algorithms. We first analyze the resulting quality of each

sampling method: the non-reuse sampling, histogram-based reuse sampling, and error-based reuse sampling. After using linear interpolation to reconstruct the dataset from each group of samples for every time-step in each data series, we calculate the peak signal-to-noise ratio (PSNR) between each reconstructed time-step and the original data (Equation 3.2). PSNR is based on the range of the data values and the cumulative squared error between the original and reconstructed values (the mean-square error). With this metric, a higher PSNR represents better quality. For our experiments, we use the input parameters as configured in Section 4.2 and listed in Table 5.1. We test with extremely small sampling ratios, with 0.5% equating to a $100:1$ compression ratio, as our sampling method is designed to maintain data fidelity with a large amount of reduction. Figure 6.4 shows the resulting quality from our experiments with varying sample ratios. Regardless of sample ratio and dataset, we find that our spatiotemporal reuse methods consistently achieve higher levels of quality than the original non-reuse sampling method. Our method is designed to never yield a quality that is less than the original sampling method, as even with the worst case where no regions are determined fit for reuse, the base algorithm is still used. This means that even in the case where no regions are ever reused, the lowest quality our algorithm will ever achieve is equivalent to the highest quality the base algorithm can achieve.

When comparing our two methods of reuse, histogram-based and error-based, we find that they yield similar levels of quality, with error-based yielding slightly higher on average. This is heavily due to the fact that the error-based method reuses regions based on the specified error tolerance; therefore, it lets in less error on average than the histogram-based method. Datasets like ExaAM consist of a central region of interest with a lot of static background data. While these static data distributions may also correlate to low levels of error, we find that the histogram similarity metric reuses more regions of data than the error based method, as shown in Figure 6.5a. For this particular dataset, reusing the regions specified by the histogram metric resulted in higher quality than the error metric.

Datasets like Hurricane Isabel consist of a region of interest, but the rest of the data is not static. The hurricane eye is the region of interest, as it has the levels of lowest pressure, but the surrounding areas also have fluctuating pressure over time. Thus, there is a lower correlation between finding similar histograms and low error across time. Figure 6.5b shows that the error based method reused more regions, however it kept the data within the error tolerance. Thus, the resulting average quality displayed in Figure 6.4b for the error-based reuse method is the highest.



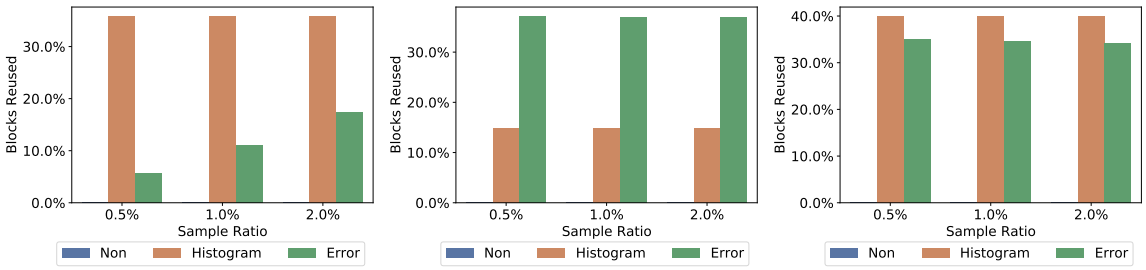
(a) Average Overall ExaAM (b) Average Overall Isabel (c) Average Overall Asteroid

Figure 6.4: Average PSNR over varying sample rates.

Lastly, we analyze the resulting quality for the asteroid impact dataset. We find the most improvement over the non-reuse method with this dataset. However, the overall PSNR for this dataset is lower than the averages of the previous two datasets. This is an artifact from the core sampling algorithm. Our spatiotemporal method improves the base group of samples, but the resulting quality is heavily limited by the performance of the base. This trend is further explored in Section 6.3.

We also include error bars of the standard deviation between the quality of each time-step, per data series. Our spatiotemporal reuse methods have degrees of high variation partially due to the non-reuse method used to gather samples. Since the base algorithm introduces a high variance in quality between time-steps, our reuse method reflects this variance. If we change the base method of choosing samples before reusing, the variance can be lowered, as further explained in Section 6.3. The second factor that introduces variance in quality is the number of regions that are reused per time-step. As the data series fluctuates, some neighboring time-steps may be very similar to each other, yielding high levels of reuse and higher quality. However, there may be parts of the series where the dataset is no longer smooth, and the simulation changes rapidly, yielding lower levels of similarity across neighboring time-steps and lower quality for those time-steps.

Next, we analyze the average throughput of each sampling method. Overall, our spatiotemporal sampling method is generally bounded by the minimum throughput of the base method of gathering samples. This trend is shown in Figure 6.6, as we detail the throughput (MB/s) of each sampling algorithm. Since we add the process of checking similarity between time-steps, the fastest our algorithm can be is the slowest the base method is. This trend is further explored in Section 6.3. However, the cost of checking for similarity is offset if the number of regions to be reused is high.



(a) Average Overall ExaAM (b) Average Overall Isabel (c) Average Overall Asteroid

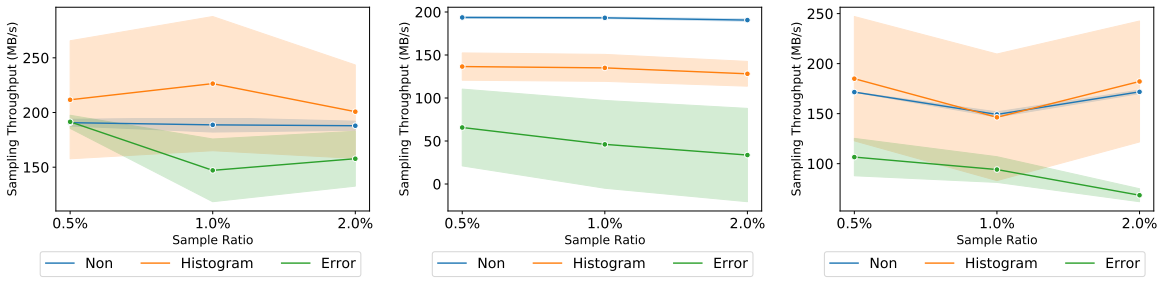
Figure 6.5: Average percentage of regions reused over varying sample rates.

When a region is reused, we do not take new samples for that data; we reuse the samples from the previous time-step. Therefore, we do not have to run the costly sampling algorithm to take new samples. If the number of reused regions is high, then the overall throughput is increased. This trend is found in Figure 6.5a in correlation to the resulting throughputs in Figure 6.6a with the histogram-based reuse method. This trend is also partially seen with the Impact dataset. However, with the Isabel dataset, the histogram-based method reuses fewer regions, so the histogram similarity overhead is not offset.

When analyzing the difference in throughput between the histogram-reuse and error-reuse methods, we find that overall, the error-based reuse method is the slowest. This is primarily due to step 5 (Region Comparison and Reuse). This step is slower for the error similarity metric than the histogram metric, as seen in Figure 6.7. The main slowdowns of our spatiotemporal sampling method, regardless of similarity metric, are found in steps 5 (Region Comparison and Reuse) and 7 (Gathering Sample Data) as shown in Figure 6.7 and described in Section 4.3. Step 5 can be trivially parallelized, as each region can be compared to its corresponding chronological neighbor independently of all of the other regions. Doing so would result in no effect on quality, but a significant increase in throughput. Step 7 can also be parallelized, where each sample is gathered from the selection specified in the previous step. Each sample value and location in the global array index can be gathered independently, concurrently.

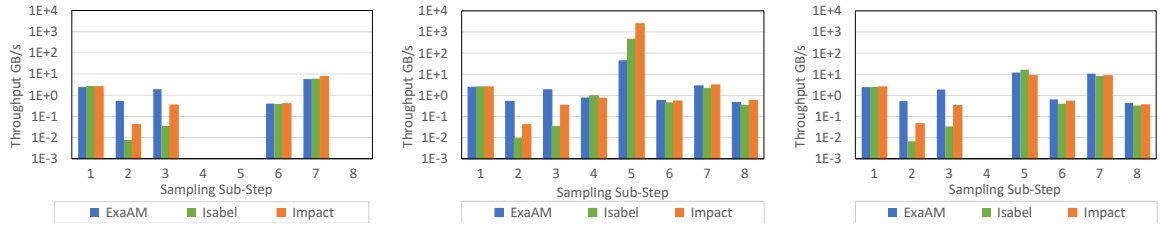
6.3 Core Sampling Algorithm Exchange

The novelty of our spatiotemporal sampling method is that it takes some samples from the current time-step in some regions but has the ability to reuse samples from previous time-steps in



(a) ExaAM. (b) Isabel. (c) Impact.

Figure 6.6: Average bandwidth of sampling process.

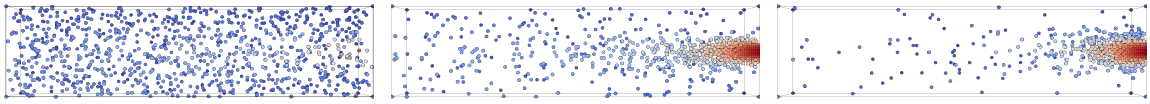


(a) Non-Reuse. (b) Histogram Reuse. (c) Error Reuse.

Figure 6.7: Average bandwidth of sampling sub-processes.

other regions, capitalizing on similarities between chronological neighbors. The process by which we gather these samples, however, is independent of our sampling method. We refer to this base method as the "Core Sampling Algorithm" (CSA). Our spatiotemporal sampling method uses the CSA when gathering samples for the first time-step and in regions that do not reuse previous samples.

Since our method works independently of CSA, we have the ability to change the CSA to any existing sampling algorithm. Thus giving our method the ability to maintain relevance as the data reduction field grows. To show the variance and usefulness of this ability, we use our histogram-based and error-based reuse methods with three different CSAs: Simple Random, Importance-Based [7], and Multi-Criteria Importance-Based Sampling [6]. Simple random sampling selects a data point into the sample by pure randomness. Importance-Based sampling uses the distribution of data values to give a bias to more unique data values. Multi-Criteria Importance-Based sampling uses the distribution of values and local gradient to give a bias based on rare value and abrupt change. The differences in the locations of samples with a 0.5% sample ratio can be seen in Figure 6.8. Simple random sampling produces a uniformly distributed sample set, while the importance-based methods bias the rare values in the region of interest. The multi-criteria sampling method yields the



(a) Random Samples (b) Importance-Based Samples (c) Multi-Criteria Samples

Figure 6.8: ExaAM samples gathered by different sampling algorithms

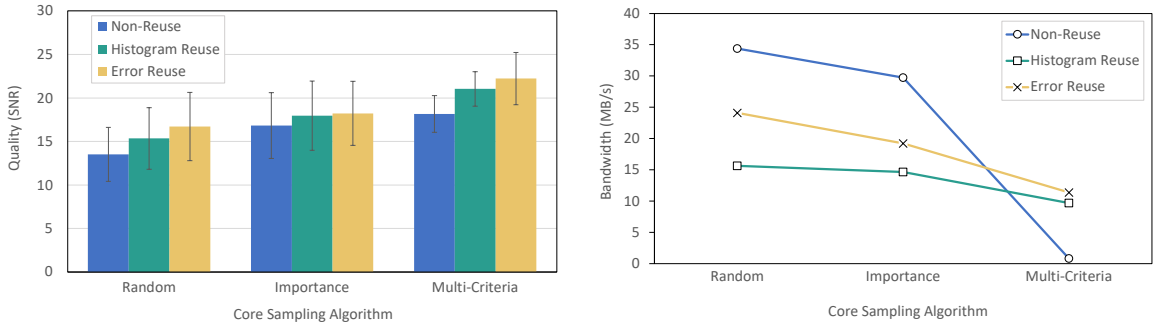
most samples in the region of interest, as it gives a bias to data points that have both rare values and abrupt changes in gradients. The location of samples heavily affects the post-reconstruction quality. Since the regions of interest have the most variance, they need more samples to ensure a higher level of quality. However, fewer data points outside of this region need to be saved, as they can be sufficiently represented by a fewer number and still maintain relatively high quality. Thus, in general, the more samples taken in the region of interest, the higher the overall average post-reconstruction quality. This trend is seen in Figure 6.9a, in relation to Figure 6.8, where the multi-criteria method has the most samples in the region of interest and, in turn, has the highest average quality. We use signal-to-noise ratio (Equation 6.1) for this evaluation to be consistent with their previous results. We also include the error bars as the standard deviation of the difference in quality between each time-step. The amount of variance seen in these error bars is dependent on the CSA used. The non-reuse version of multi-criteria sampling has a lower variance than the other methods; thus, our spatiotemporal reuse methods also have lower variance.

$$SNR = 20 * log_{10} \frac{\sigma_{raw}}{\sigma_{noise}} \tag{6.1}$$

Even though the multi-criteria algorithm yields the highest quality, it is not necessarily the best overall algorithm. In general, more sophisticated algorithms yield a better sample, but at the cost of more data assessment. Thus, the better the quality of samples, the lower the throughput. This trade-off is presented in Figure 6.9b, whereas the sampling process becomes more rigorous, the slower the overall algorithm.

6.4 Summary

In this section, we have shown that we enable higher performance by using our configuring pre-processing step. This is a critical step in the overall process, as the number of bins, error



(a) Quality.

(b) Bandwidth.

Figure 6.9: ExaAM evaluation of varying core sampling algorithms

threshold, and region size heavily affect both the quality and throughput of each sampling method. By introducing the slight temporal overhead of the steps to configure the input parameters, both quality and throughput can be closer to what the user wants for their situation. Overall, our spatiotemporal sampling algorithm yields higher quality than other sampling methods at the same reduction rate. However, it does so at the cost of some loss in throughput. Lastly, we show that our algorithm can be applied to any existing and future sampling methods, allowing our method to stay relevant as the field of data sampling grows.

Chapter 7

Conclusions and Future Work

7.1 Conclusions and Contributions

The improvements in High-Performance computation power yield larger, more detailed simulations and datasets. Since the storage size and transfer speeds have not improved at the same rate, there is a bottleneck in the overall scientific simulation pipeline. To avoid this bottleneck, others have studied, analyzed, and presented various forms of reducing the data before writing to memory. Existing work has previously studied the process of reducing the size of data by leveraging spatial or temporal redundancies within the data series. However, few works combine both aspects of the data to utilize the most redundancy possible. This research aimed to identify and leverage spatial and temporal redundancies within a data series to yield higher post-reconstruction quality at the same rate of reduction. We show that by reusing samples from neighboring time-steps in certain regions, we achieve improvement in quality both in the overall dataset and within the regions of interest. We study how various configurations of our spatiotemporal sampling algorithm, including the similarity metric, number of bins, error threshold, and region size, can affect the overall results. Based on the criticalness of these parameters, we also design and provide a set of pre-processing steps to aid the user in selecting an optimal input configuration to yield the levels of quality and/or throughput they desire. We also study the ability to interchange the base sampling algorithm that our method uses to gather samples such that any existing or future algorithm can be enhanced by appending our spatiotemporal reuse method.

7.2 Theoretical Implications

The major contribution of this research was to provide a data reduction algorithm that improves the post-reconstruction quality of existing data sampling methods. We accomplish this by introducing a temporal reuse aspect that can be applied to any other form of data sampling. The process of analyzing time-steps to find temporal redundancies by use of histograms or error can enhance any sampling algorithm. Specifically, we find our method achieves a 31.3% higher post-reconstruction quality while only introducing a 37% degradation in throughput, on average. Thus, as future sampling algorithms become faster or achieve higher qualities, our spatiotemporal method can continue to be applied, allowing us to maintain relevance as the field grows. By creating a data reduction method that is very customizable to specific constraints, we are able to greatly reduce the size of large datasets while maintaining higher levels of quality than the initial base sampling algorithm used. This improves the overall field of data sampling and makes any algorithm more efficient in terms of quality. Thus, our spatiotemporal sampling method is important in solving the issue of reducing the size of data to avoid the I/O and storage bottlenecks without loss of data fidelity.

7.3 Future Work

An area for future work includes further investigations of other aspects of the data to leverage. Currently, our method finds redundancies in the data values relative to their spatial and temporal locations. We utilize these redundancies to further reduce the size of data. However, other possible aspects of the data series that can be exploited, including inter-variable relationships. For example, there is a strong correlation between the variables in the Hurricane Isabel dataset, specifically within the region of interest. In the hurricane eye, the pressure is the lowest, and the wind speeds are low. We can use the correlation between these variables to reuse more information, like the locations of samples, to introduce more reduction.

Our method currently leverages two forms of existing data reduction schemes: data sampling and temporal selection. It reduces data by gathering a selection of samples that represent each time-step. This list of sample values is a highly lossily compressible list of floats and their locations are a list of highly losslessly compressible integers. Thus, instead of competing with other forms of data reduction, we can leverage compression to further reduce the size of the data while maintaining the

same level of data quality.

Another suggestion for future work has a focus on the reconstruction portion of the overall process. In this work, we use generic linear interpolation to convert the samples to full resolution. Even though this process yields higher quality than other methods, like nearest neighbors, it suffers from lower throughputs. As this is an equally important part of the process as the reduction step, it needs to have a more in-depth study. Creating a reconstruction algorithm specific to our sampling process may yield a resulting quality equivalent to or higher than currently available and may also address the throughput constraints.

Bibliography

[1] Top500 list. `https://top500.org/lists/top500/2021/06/`, 2021.

[2] Hiroshi Akiba, Nathaniel Fout, and Kwan-Liu Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *EuroVis*, volume 6, pages 1–8, 2006.

[3] Hiroshi Akibay and Kwan-Liu May. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of the 9th Joint Eurographics/IEEE VGTC conference on Visualization*, pages 115–122, 2007.

[4] Allison H Baker, Haiying Xu, John M Dennis, Michael N Levy, Doug Nychka, Sheri A Mickelson, Jim Edwards, Mariana Vertenstein, and Al Wegener. A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 203–214, 2014.

[5] James Belak, John Turner, and ExaAM Team Team. Exaam: Additive manufacturing process modeling at the fidelity of the microstructure. *APS*, 2019:C22–010, 2019.

[6] Ayan Biswas, Soumya Dutta, Earl Lawrence, John Patchett, Jon C. Calhoun, and James Ahrens. Probabilistic data-driven sampling via multi-criteria importance analysis. *IEEE Transactions on Visualization and Computer Graphics*, page 1–1, 2020.

[7] Ayan Biswas, Soumya Dutta, Jesus Pulido, and James Ahrens. In situ data-driven adaptive sampling for large-scale simulation data summarization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization - ISAV '18*, page 13–18. ACM Press, 2018.

[8] Franck Cappello, Sheng Di, Sihuan Li, Xin Liang, Ali Murat Gok, Dingwen Tao, Chun Hong Yoon, Xin-Chuan Wu, Yuri Alexeev, and Frederic T Chong. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications*, 33(6):1201–1220, 2019.

[9] S. Di and F. Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, page 730–739, May 2016.

[10] Paul Daniel Dumitru, Marin Plopeanu, and Dragos Badea. Comparative study regarding the methods of interpolation. *Recent advances in geodesy and Geomatics engineering*, 1:45–52, 2013.

[11] Veronika Eyring, Sandrine Bony, Gerald A Meehl, Catherine A Senior, Bjorn Stevens, Ronald J Stouffer, and Karl E Taylor. Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958, 2016.

[12] Yi Gu and Chaoli Wang. Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, 2011.

[13] Salman Habib, Vitali Morozov, Nicholas Frontiere, Hal Finkel, Adrian Pope, and Katrin Heitmann. Hacc: extreme scaling and performance across diverse architectures. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–10. IEEE, 2013.

[14] James W Hurrell, Marika M Holland, Peter R Gent, Steven Ghan, Jennifer E Kay, Paul J Kushner, J-F Lamarque, William G Large, D Lawrence, Keith Lindsay, et al. The community earth system model: a framework for collaborative research. *Bulletin of the American Meteorological Society*, 94(9):1339–1360, 2013.

[15] Hurricane ISABEL Simulation Data. `http://vis.computer.org/vis2004contest/data.html`, 2019. Online.

[16] Rob J Hyndman. The problem with sturges' rule for constructing histograms. *Monash University*, pages 1–2, 1995.

[17] Zach Jibben. truchas-pbf. `https://gitlab.com/truchas/truchas-pbf/`, 2020.

[18] Sian Jin, Pascal Grosset, Christopher M Biwer, Jesus Pulido, Jiannan Tian, Dingwen Tao, and James Ahrens. Understanding gpu-based lossy compression for extreme-scale cosmological simulations. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 105–115. IEEE, 2020.

[19] James P Kossin and Wayne H Schubert. Mesovortices in hurricane isabel. *Bulletin of the American Meteorological Society*, 85(2):151–153, 2004.

[20] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018.

[21] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, Dec 2014.

[22] Aidong Lu and Han-Wei Shen. Interactive storyboard for overall time-varying data visualization. In *2008 IEEE Pacific visualization symposium*, pages 143–150. IEEE, 2008.

[23] William G Madow and Lillian H Madow. On the theory of systematic sampling, i. *The Annals of Mathematical Statistics*, 15(1):1–24, 1944.

[24] Dirk Meister, Jurgen Kaiser, Andre Brinkmann, Toni Cortes, Michael Kuhn, and Julian Kunkel. A study on data deduplication in hpc storage systems. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

[25] Sheri Mickelson, Alice Bertini, Gary Strand, Kevin Paul, Eric Nienhouse, John Dennis, and Mariana Vertenstein. A new end-to-end workflow for the community earth system model (version 2.0) for the coupled model intercomparison project phase 6 (cmip6). *Geoscientific Model Development*, 13(11):5567–5581, 2020.

[26] Bao D Nguyen, Ngan VT Nguyen, Vung Pham, and Tommy Dang. Visualization of data from hacc simulations by paraview. In *2019 IEEE Scientific Visualization Conference (SciVis)*, pages 31–32. IEEE, 2019.

[27] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens. Adr visualization: A generalized framework for ranking large-scale scientific data using analysis-driven refinement. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 43–50, 2014.

[28] Palmetto Cluster, Clemson University. `http://citi.clemson.edu/palmetto/`, 2021. [Online; accessed 24-June-2021].

[29] John Patchett and Galen Gisler. Deep water impact ensemble data set. Technical report, Los Alamos National Laboratory, 2017. LA-UR-17-21595.

[30] Veronika Solteszova, Noeska N Smit, Sergej Stoppel, Renate Grüner, and Stefan Bruckner. Memento: Localized time-warping for spatio-temporal selection. In *Computer Graphics Forum*, volume 39, pages 231–243. Wiley Online Library, 2020.

[31] Seung Woo Son, Zhengzhang Chen, William Hendrix, Ankit Agrawal, Wei-keng Liao, and Alok Choudhary. Data compression for the exascale computing era-survey. *Supercomputing frontiers and innovations*, 1(2):76–88, 2014.

[32] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, page 1129–1139, May 2017.

[33] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1129–1139. IEEE, 2017.

[34] KE Taylor, RJ Stouffer, GA Meehl, KE Taylor, RJ Stouffer, and GA Meehl. An overview of cmip5 and the experiment design, b. am. meteorol. soc., 93, 485–498, 2012.

[35] S Crusan V Vishwanath and K Harms. Parallel i/o on mira, 2019.

[36] Tzu-Hsuan Wei, Soumya Dutta, and Han-Wei Shen. Information guided data sampling and recovery using bitmap indexing. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 56–65. IEEE, 2018.

[37] D Whalen and ML Norman. Competition data set and description. *2008 IEEE Visualization Design Contest*, 2008.

[38] Jonathan Woodring, J Ahrens, J Figg, Joanne Wendelberger, Salman Habib, and Katrin Heitmann. In-situ sampling of a large-scale particle simulation for interactive visualization and analysis. In *Computer Graphics Forum*, volume 30, pages 1151–1160. Wiley Online Library, 2011.

[39] Bo Zhou and Yi-Jen Chiang. Key time steps selection for large-scale time-varying volume datasets using an information-theoretic storyboard. In *Computer Graphics Forum*, volume 37, pages 37–49. Wiley Online Library, 2018.