Research Article

BULLETIN OF COMPUTER AND DATA SCIENCES

Adaptive Multi-Resolution Stratified Sampling for Interactive Visualization of Massive Time Series

William W. Predebon

Retired professor and chair of the Department of Mechanical Engineering-Engineering Mechanics, at Michigan Technological University

Abstract

Massive time-series datasets from sensor networks, scientific monitoring, and online services pose a challenge for interactive visual exploration, as the number of samples can exceed screen pixels by several orders of magnitude. Naïve subsampling harms interpretability and event detection, and existing pixel-aware methods usually operate at a single resolution, failing to exploit the multi-resolution nature of zoom and pan workflows. We propose *Adaptive Multi-Resolution Stratified Sampling* (AMRSS), which combines stratified min–max sampling with a hierarchical multi-resolution index and an adaptive query-time selection algorithm. AMRSS precomputes a summary tree whose nodes store min–max envelopes and a local variability score; for a given viewport and pixel budget, a top-down traversal selects variable-resolution segments that concentrate samples in visually complex regions while avoiding oversampling in flat regions, yielding a sample set bounded by the screen width and renderable as a polyline with predictable complexity. We formalize pixel-aware sampling under viewport constraints, present the AMRSS design and its time–space complexity, and outline an evaluation protocol against uniform subsampling, fixed-resolution min–max, and triangle-based methods on seismic, physiological, and synthetic datasets, providing a practical basis for interactive dashboards and a framework extensible to formal error bounds, streaming updates, and perceptual studies.

Keywords: pixel-aware time-series visualization, adaptive multi-resolution sampling, stratified min–max envelopes, interactive data exploration, multi-resolution indexing

1. Introduction

Modern sensing and monitoring infrastructures continuously generate high-frequency time-series data at scales that challenge conventional visualization pipelines. Seismic observatories, industrial process monitoring systems, power-grid telemetry, financial tick streams, and large-scale web service logs all produce continuous flows of measurements that analysts routinely inspect through plots of raw or lightly processed signals to detect anomalies, compare patterns, validate models, and inform operational decision-making [1, 2]. In these settings, interactive visualization is often the first and most flexible analytical tool, yet it must operate under strict latency and resolution constraints imposed by commodity displays and web-based rendering environments.

A fundamental scale mismatch arises between the number of observed data points and the number of available screen pixels. A single day of 100 Hz measurements yields 8,640,000 samples, while a

Corresponding author: William W. Predebon (w.prebon@gmail.com)

BCDS 2024, Vol. 5(4), 60-73. DOI: 10.71448/bcds2454-5

typical display provides on the order of 10^3 horizontal pixels. Plotting every point is computationally expensive for real-time interaction and visually redundant, since many nearby samples fall into the same pixel column and cannot be distinguished. Naïve subsampling strategies, such as drawing every k-th point, reduce load but risk erasing narrow spikes, rare events, and local extrema that may be scientifically or operationally critical [3]. Shape-preserving methods like the Largest-Triangle-Three-Buckets algorithm attempt to mitigate this by selecting representative points that best capture local geometry [4], while recent min–max-based techniques summarise each window or pixel column by its extreme values to better preserve envelopes and outliers [5, 6].

These approaches are important steps toward *pixel-aware* time-series sampling, where the sample budget is explicitly tied to the screen resolution. However, most existing methods operate at a single global resolution per view and do not fully exploit the inherently multi-resolution nature of interactive workflows. Users routinely zoom into local regions, zoom out for context, and pan across time, issuing repeated, overlapping viewport queries. Multi-resolution schemes and pyramids for time-series visualization have been explored in earlier work, but they typically select a uniform resolution level for a given zoom factor, so that all parts of the visible time range are rendered at the same level regardless of local variability [7, 8]. As a result, flat regions are oversampled while highly dynamic regions may still be undersampled at coarser scales, and precomputed summaries are not always reused efficiently across overlapping viewports.

In this paper we address these limitations by introducing *Adaptive Multi-Resolution Stratified Sampling* (AMRSS), a method designed from the outset for interactive, multi-resolution usage and inspired by stratified sampling techniques for plotting massive time series [9]. AMRSS decouples preprocessing from query-time sampling through a hierarchical index over time. During preprocessing, the method constructs a binary (or b-ary) tree of segments, each summarising a contiguous time interval by its min–max values and a local variability score that approximates the visual complexity of the segment. At interaction time, given a viewport, a pixel width, and a sample budget, the system traverses this tree top-down, adaptively refining complex regions while leaving flat regions at coarser resolution. The resulting sample set respects a strict budget proportional to screen width, can be rendered as a polyline with predictable worst-case complexity, and naturally supports repeated, overlapping queries that arise when users pan and zoom.

The contributions of this work are fourfold. First, we formalize pixel-aware time-series sampling as a constrained optimisation problem over viewports, in which the objective is to maximise visual fidelity under a strict sample budget tied to the pixel width, thereby connecting visualization-oriented aggregation to explicit resource constraints [5, 6]. Second, we introduce AMRSS as a practical method that combines stratified min–max sampling with a multi-resolution summary tree and an adaptive query-time refinement strategy; this design supports arbitrary viewports, reuses precomputed summaries across overlapping queries, and can be extended to multivariate signals. Third, we analyse the time and space complexity of the approach, showing that the summary structure can be constructed in linear time with $O(N)$ storage and that viewport queries can be answered in time proportional to the sample budget times a small logarithmic factor, making the method suitable for real-time dashboards. Finally, we outline a comprehensive experimental protocol for evaluating interactive time-series sampling methods, including visual similarity metrics, event-preservation criteria, and latency measurements across seismic, physiological, and synthetic datasets, positioning AMRSS within and against existing uniform, min–max, and triangle-based baselines [3, 4, 7, 8]. Although we focus on scalar time series and polyline plots, the general design extends naturally to multivariate settings,

envelope visualizations, and streaming updates, making AMRSS a flexible building block for scalable, perceptually aware time-series visualization systems.

More sophisticated techniques seek to preserve the overall *shape* of the series under a fixed sample budget rather than simply thinning points uniformly. Triangle-based methods, such as the Largest-Triangle-Three-Buckets (LTTB) algorithm, select representative points that maximise the area of triangles formed with neighbouring samples, thereby favouring samples that contribute most to the perceived geometry of the polyline and reducing visual distortion for a given budget [3, 4]. Min–max-based schemes instead aggregate each pixel-width window to its minimum and maximum values, guaranteeing that the rendered polyline covers the full vertical range of the original series in each column and better preserving spikes and outliers that might be lost under uniform subsampling [5, 6]. These approaches have been adopted in a variety of dashboard systems and monitoring tools, where they provide a practical compromise between rendering cost and visual fidelity compared to naïve sampling or simple averaging [2].

Recent work has further explored visualization-oriented aggregation methods that provide formal guarantees on visual error or envelope coverage, often by treating time-series summarisation as an optimisation problem under pixel and latency constraints [5, 6]. Such methods typically precompute summaries at one or more resolutions—using min–max envelopes, piecewise-linear approximations, or multi-level caches—and then serve interactive queries by translating those summaries into plot-ready samples [7, 8]. However, most existing schemes operate with either a single global resolution or a small fixed set of resolutions per viewport and lack an explicit adaptive mechanism for varying resolution across different time regions within the same view. In practice, this means that flat regions are often oversampled while highly dynamic segments remain undersampled at coarser zoom levels, and overlapping viewports during pan and zoom cannot fully exploit hierarchical reuse. The AMRSS framework proposed in this paper is designed to address precisely these limitations by combining min–max stratification with a genuinely adaptive, viewport-aware multi-resolution hierarchy [9].

2. Problem Formulation

2.1. Pixel-aware sampling under viewport constraints

Let $x(t)$ be a real-valued time series observed at N discrete time points $t_1 < t_2 < \cdots < t_N$, with corresponding values $x_1, \ldots, x_N$. An interactive visualization system displays a subset of this series within a viewport

$$V = [T_{\min}, T_{\max}] \subseteq [t_1, t_N],$$

on a plotting area of width W pixels. The horizontal axis is assumed to be linear in time, and the visible interval $[T_{\min}, T_{\max}]$ is mapped to pixel columns by a function

$$\pi : [T_{\min}, T_{\max}] \to \{1, \ldots, W\},$$

which assigns each time point to a discrete pixel position. This setting captures the common case of web-based dashboards and desktop plotting libraries, where the horizontal resolution is fixed by the display and rendering engine [1, 2, 10–13].

Because each pixel column can display only a small number of visually distinguishable samples, the system can render at most B sample points within the viewport, where B is typically chosen to be proportional to W (for example, $B = 4W$ when using up to two min–max pairs per pixel column, as

in min–max-based aggregation schemes [5, 6]). This constraint reflects both display limitations and graphical perception results showing that overplotting and excessive density degrade the readability of visual encodings [14, 15]. Let $S \subseteq \{1, \ldots, N\}$ denote the indices of samples selected for display, restricted to those time points that lie within the viewport, so $t_i \in V$ for all $i \in S$. The rendered plot is then obtained by connecting the points (t_i, x_i) for $i \in S$ in order of increasing time, as in standard polyline visualizations [10].

Within this setting, pixel-aware sampling can be viewed as a constrained optimisation problem defined over the choice of index set S. The first requirement is a cardinality constraint: the sample budget must not be exceeded, so $|S| \leq B$. The second requirement is that the rendered polyline should approximate, as closely as possible, the visual appearance of the full-resolution plot of $x(t)$ within the viewport V. To make this second requirement precise, we treat the plot as a raster image and define an image-based similarity measure between the full-resolution rendering and the downsampled rendering. Possible choices include pixel-wise mean squared error, structural similarity indices, or envelope-based discrepancies that compare upper and lower envelopes across pixel columns [3, 5, 6, 11]. In practice, the exact form of this similarity measure depends on the visualization library, antialiasing settings, and display characteristics, but the general goal is to ensure that important visual features—such as spikes, local extrema, and trend changes—are preserved under the budget constraint, in line with findings from graphical perception and design principles for quantitative displays [14, 15]. From this perspective, methods like LTTB [3, 4], min–max sampling [5, 6], and related aggregation schemes [7, 8] can be understood as heuristics for approximately solving this constrained problem with different implicit choices of similarity metric, similar in spirit to approximation and aggregation approaches used in time-series visualization tools such as LifeLines, GeoTime, LiveRAC, and stacked graph techniques [16–18].

The AMRSS framework proposed in this paper adopts the same formal viewpoint, but replaces purely window-based or triangle-based selection rules with an adaptive, hierarchical mechanism. Rather than choosing S directly at the level of individual points, AMRSS operates on a multi-resolution partition of the time axis into segments, each with its own local summary and variability score [9]. For a given viewport and budget, the method selects a collection of segments whose associated min–max representatives induce a sample set S that satisfies $|S| \leq B$ while aiming to maximise visual fidelity according to the chosen similarity measure. This segment-based formulation is crucial for supporting the interactive and multi-resolution requirements discussed next and aligns with broader perspectives on time-oriented visual analysis, where data, tasks, and representations are treated jointly [10, 12].

2.2. Requirements for interactive systems

Beyond the static optimisation view, interactive visualization imposes additional constraints that strongly shape the design of a practical sampling method. First, viewport locality is essential: users typically pan and zoom in small increments, issuing many overlapping viewport queries over time. A sampling algorithm that treats each query independently will repeatedly recompute similar summaries, incurring unnecessary latency and wasted computation. Instead, the underlying data structure should allow reuse of precomputed information across neighbouring viewports, so that small viewport changes translate into small incremental updates in the sample set [2, 7, 19].

Second, multi-resolution support is required to accommodate different zoom levels. As users zoom in, the effective time range per pixel shrinks and the sample budget per viewport may increase,

allowing finer-grained representation of local detail. Conversely, zooming out increases the time span per pixel and typically forces a coarser summarisation to stay within a fixed budget or latency target. A single global resolution per zoom level, as in traditional pyramid approaches, is often insufficient because it ignores the strong heterogeneity in temporal variability across the viewport [7, 8, 11, 16]. A desirable sampling method should therefore vary resolution within the same viewport, using more segments and samples in complex regions and fewer in flat regions, while still respecting the overall budget and supporting the "overview first, zoom and filter, then details on demand" mantra for interactive visual analysis [12, 13].

Third, strict latency constraints arise from the need to maintain a smooth, responsive interactive experience. Empirical guidelines from visual analytics and human–computer interaction suggest that updates should complete within tens of milliseconds to feel fluid; longer delays break the sense of direct manipulation and can discourage exploratory analysis [1, 12]. This places tight bounds on the end-to-end computation required to answer a viewport query, including both internal sampling logic and any rendering pipeline overhead. Algorithms with super-linear behaviour in the sample budget or viewport length may struggle to meet these constraints for high-frequency time series or complex dashboards, especially in domains such as system management, finance, or large-scale telemetry where tools like LiveRAC and related interfaces have highlighted the importance of interactive performance [17].

Finally, many real-world applications involve continuously arriving data streams or extremely long-running monitoring deployments. In such settings, incremental extensibility becomes important: the sampling structure should support efficient updates as new data points are appended, without requiring full recomputation of multi-resolution summaries. This includes controlling the depth and size of hierarchical indexes, managing memory usage over time, and potentially incorporating mechanisms for aging or discarding very old data [5, 6, 10].

The AMRSS framework is specifically designed to address these interactive requirements by separating offline preprocessing, in which a multi-resolution index over time is constructed, from online query-time sampling, in which segments are adaptively selected given a viewport and sample budget. The hierarchical summary tree supports viewport locality and multi-resolution behaviour through top-down refinement and coarse-to-fine selection; its bounded size and simple statistics enable low-latency queries; and its structure lends itself to incremental updates in streaming scenarios. Together, these properties make AMRSS a suitable foundation for scalable, pixel-aware time-series visualization in modern interactive systems [9, 12].

3. Adaptive Multi-Resolution Stratified Sampling

3.1. Overview

AMRSS represents the time series via a hierarchical partition of the time axis into contiguous segments that collectively form a multi-resolution summary of the data. Each segment I is associated with the index set

$$\mathcal{I}(I) = \{i : t_i \in I\},$$

and is summarised by a small number of statistics: the minimum value within the segment and its time index, $x_{\min}(I) = \min_{i\in\mathcal{I}(I)} x_i$ with $i_{\min}(I) = \arg\min_{i\in\mathcal{I}(I)} x_i$; the maximum value and its time index, $x_{\max}(I) = \max_{i\in\mathcal{I}(I)} x_i$ with $i_{\max}(I) = \arg\max_{i\in\mathcal{I}(I)} x_i$; and a variability score $v(I)$ that quantifies the local visual complexity of the segment, for example via the range, variance, or deviation

from a simple trend. In spirit, these summaries are closely related to min–max-based aggregation schemes that have been shown to preserve the vertical envelope of time series under pixel constraints [5, 6], but here they are embedded within an explicitly hierarchical structure.

Segments are organised into a rooted tree $\mathcal{T}$, where the root covers the entire time range $[t_1, t_N]$ and each internal node is split into child segments that cover disjoint subintervals. For concreteness, we consider a binary tree with equal-length splits in the time domain, although other branching factors or split strategies (e.g., data-driven boundaries or non-uniform partitions) are possible and could be explored in future work [7, 8]. The tree provides a family of nested segmentations, ranging from coarse summaries at the root to fine-grained segments near the leaves. Given a viewport V, pixel width W, and sample budget B, the sampling algorithm traverses $\mathcal{T}$ top-down, starting from nodes whose intervals intersect V. At each node, it decides whether to *accept* the node—using its min–max summary as representatives—or to *refine* it by recursing into its children, based on the variability score, the projected pixel footprint, and the remaining sample budget. The final output is a set of accepted nodes $\mathcal{A}$, whose min–max pairs yield the sample index set S used to draw the polyline. This design combines the envelope-preserving strengths of min–max aggregation [5, 6] with the adaptive spatial allocation characteristic of multi-resolution and pyramid-based schemes [7, 8], while explicitly targeting the pixel-aware optimisation problem and the interactive constraints of modern dashboards [1, 2, 9].

3.2. Preprocessing: building the summary tree

The summary tree $\mathcal{T}$ is constructed in an offline preprocessing stage using a single pass over the time series. Let $L_{\max}$ denote the maximum depth of the tree, chosen to correspond either to a minimum segment length $\Delta t_{\min}$ or to a maximum number of points per leaf M. The construction starts by initialising the root node $I_1^{(0)} = [t_1, t_N]$ at level 0, covering the entire time axis. This root is then recursively subdivided: at each level $\ell = 0, 1, \ldots, L_{\max} - 1$, each segment $I_k^{(\ell)}$ is partitioned into two child intervals $I_{2k-1}^{(\ell+1)}$ and $I_{2k}^{(\ell+1)}$ of equal time length by splitting at the midpoint of its time range. Alternative partitioning strategies, such as splitting at quantiles of data density or aligning boundaries with domain-specific events, could be incorporated without altering the core algorithmic ideas [7, 8].

Once the topology of the tree is fixed, each data point (t_i, x_i) is streamed through the structure and assigned to all nodes whose intervals contain t_i along the path from root to leaf. As points are processed, each node I updates its summary statistics. The minimum and maximum values, $x_{\min}(I)$ and $x_{\max}(I)$, are obtained by tracking the extremal values of x_i over $i \in \mathcal{I}(I)$, and the corresponding indices $i_{\min}(I)$ and $i_{\max}(I)$ are recorded to preserve temporal information. To capture local visual complexity, a variability score $v(I)$ is computed; a simple choice is the range-based score

$$v_{\text{range}}(I) = x_{\max}(I) - x_{\min}(I),$$

which emphasises segments with large vertical excursions. To better account for shape, one can approximate the error incurred by representing the segment with a straight line between its endpoints. Let (t_a, x_a) and (t_b, x_b) be the first and last time points in $\mathcal{I}(I)$ and let $x_{\text{lin}}(t)$ denote their linear interpolation. A deviation-based score is then defined as

$$v_{\text{dev}}(I) = \max_{i \in \mathcal{I}(I)} |x_i - x_{\text{lin}}(t_i)|,$$

or via an aggregate measure such as mean absolute deviation. This mirrors ideas from piecewise-linear approximation and simplification methods used in time-series compression and visualisation [3, 4], but here the score is used solely as a priority signal for adaptive refinement.

In practice, the tree can be constructed in $O(N)$ time by incrementally updating statistics at each node as samples are streamed, without repeatedly rescanning the data. The total number of nodes in a complete binary tree with $L_{\max}+1$ levels is $O(2^{L_{\max}})$; if $L_{\max}$ is set so that leaves contain $O(1)$ data points on average, this scales as $O(N)$ in both time and space. This linear behaviour is comparable to that of recent min–max caching and aggregation schemes [5, 6], but the hierarchical arrangement enables the adaptive query-time behaviour that distinguishes AMRSS from purely level-based pyramids.

3.3. Query-time sampling: adaptive refinement

At query time, given a viewport $V = [T_{\min}, T_{\max}]$, pixel width W, and sample budget B, the AMRSS sampling algorithm operates on the precomputed tree $\mathcal{T}$ to select a set of accepted nodes $\mathcal{A}$ whose summaries induce a sample set S satisfying $|S| \leq B$. The procedure begins by identifying the nodes at the highest level (closest to the root) whose intervals intersect V; these nodes form the initial frontier $\mathcal{F}$ from which refinement proceeds. For each node I in the frontier, three quantities guide the decision: the projected pixel span of I, defined as

$$\mathrm{pix}(I) = \left|\{p \in \{1, \ldots, W\} : \exists t \in I \cap V \text{ with } \pi(t) = p\}\right|,$$

the variability score $v(I)$, and the remaining sample budget B_{rem}, initially set to B and decremented as nodes are accepted.

Intuitively, segments that are both visually complex (high $v(I)$) and wide in pixel space (large $\mathrm{pix}(I)$) are prime candidates for refinement, while flat or narrow segments can be safely represented at coarser resolution. Accepting a node consumes up to four sample points (two min–max pairs) for its pixel span, depending on how the extremal points map to pixel columns. For conservative budgeting, this cost can be approximated as a small constant $c(I)$ (e.g., 2 or 4) per node. AMRSS maintains a priority ordering over the frontier, typically prioritising nodes with larger variability scores or larger ratios $v(I)/\mathrm{pix}(I)$, and repeatedly selects the highest-priority node for consideration. If accepting this node would not exceed the remaining budget, and if one of several stopping conditions holds—such as the node being a leaf, the variability score falling below a threshold τ, or the segment mapping to at most a single pixel column—then the node is accepted, added to $\mathcal{A}$, removed from the frontier, and its cost is subtracted from B_{rem}. Otherwise, if the node has children and can be further refined, it is replaced in the frontier by its children, thereby allocating more detailed representation to that time region. If a leaf node is reached whose cost would exceed the remaining budget, the algorithm may terminate early or invoke an optional post-processing step that coarsens some previously accepted nodes to satisfy the constraint.

This top-down, budget-aware refinement strategy is conceptually similar to adaptive mesh refinement in numerical methods and multi-resolution selection in pyramid-based visualization [7, 8], but tailored specifically to min–max summaries and the pixel-aware sampling objective. The final sample index set S is obtained by collecting $i_{\min}(I)$ and $i_{\max}(I)$ for all $I \in \mathcal{A}$ and restricting these indices to those whose time stamps fall within the viewport V. The resulting polyline can be rendered directly using standard plotting libraries and respects the strict cardinality constraint, while adaptively concentrating samples in visually complex regions.

3.4. Complexity analysis

Let N be the number of time points in the series and B the sample budget for a given viewport. Under reasonable assumptions on the branching factor and depth of $\mathcal{T}$, the preprocessing cost of constructing the tree and computing all node summaries is $O(N)$ and the storage cost is likewise $O(N)$, as each data point contributes to at most $O(\log N)$ nodes along the path from root to leaf but statistics can be aggregated in a single pass. For a fixed viewport, the number of nodes whose intervals intersect V at each level is bounded by a constant factor times the branching factor, and over the course of the algorithm the frontier $\mathcal{F}$ contains $O(B)$ nodes, because each accepted node consumes a positive portion of the budget and each refinement step replaces one node with a small, constant number of children.

Selecting the next node to refine or accept can be implemented via a priority queue keyed by variability scores or combined scores such as $v(I)/\mathrm{pix}(I)$, yielding $O(\log|\mathcal{F}|)$ per extraction. Since $|\mathcal{F}|= O(B)$, the total query time for a viewport is $O(B \log B)$ in the worst case, with a modest constant factor. In typical interactive scenarios, where the budget B is proportional to the pixel width W (e.g., a few thousand samples at most), this complexity is compatible with latency requirements for real-time dashboards and comparable to or better than the costs reported for fixed-resolution min–max sampling and related aggregation methods [5, 6]. Moreover, because AMRSS reuses the same tree across many overlapping viewports, amortised costs over sequences of pan and zoom operations can be significantly lower than those of methods that recompute window-based summaries from scratch for each query [2, 7].

3.5. Extensions

For multivariate series $(x^{(1)}(t), \ldots, x^{(d)}(t))$, AMRSS can be extended by either building a separate tree per channel or constructing a joint tree in which variability scores aggregate information across channels, for instance using the maximum of channel-wise ranges or a norm over vector-valued deviations. Query-time sampling then operates on the joint tree, and the accepted segments are applied consistently across all channels, ensuring aligned time segmentation and preserving cross-channel relationships, which is particularly important in domains such as seismology, physiological monitoring, and multi-sensor industrial systems [2].

In streaming scenarios, where new data points arrive continuously, the tree can be maintained incrementally by appending new leaves and updating statistics along the path from these leaves to the root. When the total time range grows beyond the initial design, periodic rebalancing or pruning strategies can be employed to control the depth and memory footprint of the structure, for example by merging very old segments into coarser summaries or by maintaining a sliding window of recent data. These ideas connect AMRSS to streaming and online aggregation techniques used in time-series databases and monitoring platforms [5, 6].

The choice of variability score $v(I)$ is a key design degree of freedom. Beyond simple range- or deviation-based scores, one could incorporate perceptual models that weight high-frequency variations differently from low-frequency trends, emphasise anomalies and outliers, or approximate human judgements of visual salience. Such scores might be derived from local spectral content, wavelet coefficients, or learned surrogate models trained on user studies or task-based performance data [1, 2]. Integrating these perceptually motivated scores within AMRSS would allow the refinement process to prioritise segments that matter most for human interpretation, further aligning the sampling mechanism with downstream analytical tasks and offering an avenue for future extensions of the framework

[9].

4. Experimental Design

In this section we outline an experimental protocol to evaluate AMRSS against existing time-series sampling methods, with a particular focus on visual fidelity, event preservation, and interaction latency. The design is intended to be generic enough to apply across multiple domains, yet concrete enough to reveal the strengths and limitations of AMRSS relative to widely used baselines such as uniform subsampling, triangle-based selection, and fixed-resolution min–max aggregation [3–6].

4.1. Datasets

A comprehensive evaluation should draw on datasets with diverse temporal dynamics and domain characteristics, reflecting the variety of real-world monitoring and analysis scenarios in which interactive time-series visualization is employed [1, 2]. To this end, we consider high-frequency seismic waveforms containing a mixture of background noise and transient events such as earthquakes, microseismic swarms, and anthropogenic signals, mirroring the operational setting in which stratified min–max approaches have previously been deployed [9]. We complement these with physiological signals, including electrocardiogram (ECG) and electroencephalogram (EEG) records that exhibit periodic structure, sharp spikes, artefacts, and slow drifts; such signals provide a stringent test of envelope preservation and local shape fidelity. Industrial or infrastructure telemetry, such as power consumption, server CPU utilisation, or network traffic time series, contributes examples with strong daily cycles, bursts, and regime shifts that are typical of large-scale monitoring dashboards. Finally, synthetic data are generated to systematically explore edge cases: controlled mixtures of smooth trends, oscillatory components, and injected spikes or anomalies allow us to vary characteristics such as noise level, event frequency, and temporal localisation in a controlled manner, providing an interpretive complement to real-world datasets [3, 4]. For each dataset type, the time series are partitioned into multiple windows of varying lengths, ranging from short intervals (e.g., 10 seconds) to long spans (e.g., 24 hours), in order to emulate typical exploration scenarios at different zoom levels.

4.2. Baseline methods

To place AMRSS in context, we compare it against several representative sampling strategies that capture the main design choices in the literature. As a reference for visual fidelity and latency, we include the full plot that renders all points within the viewport; although impractical for very large datasets, this serves as an upper bound on visual accuracy and a lower bound on speed. Uniform subsampling, in which every k-th point is plotted with k chosen so that the resulting sample set respects the budget B, represents the simplest baseline and is widely used in practice, despite its known tendency to miss narrow spikes and local extrema [3]. A fixed-resolution min–max baseline divides the visible time range into equal-width bins that map directly to pixel columns and represents each bin by its minimum and maximum values, following the min–max aggregation schemes proposed for scalable dashboard rendering [5, 6]. Finally, a triangle-based sampling baseline employs a Largest-Triangle-Three-Buckets-type algorithm that selects representatives based on geometric criteria under the same sample budget, favouring points that contribute most to the perceived shape of the polyline [3, 4]. All methods, including AMRSS, are implemented within the same visualization environment and rendering stack, ensuring that timing comparisons reflect differences in sampling logic rather

than external factors such as graphics backends or network overhead.

4.3. Evaluation metrics

We evaluate each method along three primary dimensions aligned with the problem formulation: visual similarity, event preservation, and interaction latency. To assess visual similarity, for each viewport we render both the full plot and the sampled plot to raster images at the same resolution and compute quantitative image-based metrics. These include pixel-wise mean squared error (MSE) between grayscale images, which captures overall luminance differences; the structural similarity index (SSIM), which is more closely aligned with human perceptual judgements of contrast and structure; and an envelope discrepancy measure defined as the average vertical distance between the upper and lower envelopes of the sampled plot and those of the full plot across pixel columns, extending envelope-based evaluation ideas from min–max sampling work [5, 6].

Event preservation is evaluated on datasets with annotated events, such as seismic phases, ECG beats, or synthetic spikes. For each annotated event interval, we check whether the sampled plot retains a visually recognisable signature of the event. A simple operational proxy is to test whether, within the event interval, the sampled plot attains at least one local extremum within a specified amplitude tolerance and temporal tolerance of the original extremum; additional criteria, such as preserved onset time or approximate width, can be used for sensitivity analysis [2, 3]. The fraction of events that satisfy these conditions summarises how well each method preserves features that are likely to be of direct analytical interest, beyond global image similarity.

Interaction latency is measured by simulating realistic interactive usage patterns consisting of sequences of pan and zoom operations. For each method, we issue a series of viewport queries that emulate horizontal panning over long time ranges and zooming in and out around points of interest, and we record end-to-end query latency from viewport specification to rendered image. To obtain stable estimates, each query is repeated multiple times, and we report distributional statistics such as median latency, 95th percentile, and worst-case latency over the sequence. These measurements connect directly to the responsiveness requirements discussed in the problem formulation and in prior work on interactive visual analytics [1, 7, 8].

4.4. Experimental protocol

For each combination of dataset and sampling method, the experimental protocol proceeds in a systematic manner that reflects typical user interactions while enabling controlled comparisons. We first define a grid of viewports that includes both narrow and wide windows over the time axis, as well as a range of zoom levels corresponding to different pixel widths W and associated sample budgets B. For every viewport and zoom configuration, we run each method to compute the corresponding sample set under its allocated budget, then render both the sampled plot and the full plot using the shared visualization environment. Visual similarity metrics and, where applicable, event preservation scores are computed from these renderings. Latency measurements are taken by timing the end-to-end process of sampling and rendering for each method, averaging over repeated runs to reduce the impact of transient system noise. The resulting measurements are summarised in tables and plots that show similarity as a function of budget, latency as a function of budget and viewport size, and trade-off curves that highlight the relative performance of AMRSS versus the baselines across different regimes. This design follows and extends evaluation strategies used in prior studies of time-series sampling and min–max aggregation for visualization [3, 5–8], while explicitly aligning the analysis

with the pixel-aware and interactive objectives that motivate AMRSS.

5. Discussion

AMRSS is designed to exploit two fundamental properties of time-series visualization that are well documented in the visual analytics and time-series aggregation literature. First, the effective resolution of any display is bounded by the number of pixels, so attempting to render substantially more than a small constant number of samples per pixel column is inherently redundant and can even reduce readability through overplotting [1, 2]. Second, visual interest is highly non-uniform over time: sharp transients, edges, oscillations, and regime changes tend to be concentrated in specific regions, while other portions of the series may be nearly flat or dominated by low-amplitude noise. Traditional uniform subsampling ignores both facts, wasting budget in visually uninformative regions and risking the removal of rare but important events [3]. Fixed-resolution min–max aggregation and triangle-based methods partially address these issues by tying the sampling budget to pixel columns and by prioritising shape-defining points [3–6], but they remain largely homogeneous within a viewport and do not fully exploit within-view heterogeneity in variability.

By decoupling preprocessing from query-time sampling and introducing an explicitly adaptive, viewport-dependent selection mechanism, AMRSS seeks to align computational effort with perceptual importance more closely than existing approaches. The hierarchical summary tree provides a compact multi-resolution representation in which each segment carries both envelope information and a variability score, allowing the query-time algorithm to refine only those regions that warrant additional samples. In concept, this mirrors the logic of multi-resolution pyramids and min–max caching schemes that precompute summaries at several scales [5–8], but AMRSS differs by mixing resolutions within a single viewport and enforcing a global budget constraint across all segments. As a result, more of the budget is spent on visually complex portions of the signal, while flat regions are represented at coarser resolution without violating the pixel-aware constraints formulated. A comprehensive empirical evaluation following the protocol would be expected to show that, for a fixed budget, AMRSS can achieve lower envelope discrepancy and better event preservation than uniform or single-resolution schemes, especially on datasets with strong temporal heterogeneity.

The hierarchical index is also central to achieving responsive interaction in practice. Because the same tree underlies many overlapping viewports, AMRSS can reuse summaries efficiently as users pan and zoom. A small horizontal pan changes the viewport boundaries only slightly, so most previously accepted nodes remain valid and only a fringe of segments near the edges must be reconsidered. Zooming in can be interpreted as increasing the effective budget and refining the selection within previously accepted segments, while zooming out corresponds to coarsening the representation by replacing groups of fine segments with their ancestors in the tree. This coarse-to-fine and fine-to-coarse behaviour parallels ideas from progressive refinement and multi-resolution exploration in other visual data exploration systems [1, 7, 8], but here it is tightly coupled to a min–max-based summarisation that is known to preserve critical envelope features under pixel constraints [5, 6]. In real deployments, such incremental updates can substantially reduce amortised latency compared to methods that recompute window-based summaries or LTTB-style selections from scratch for each viewport [2–4].

Despite these advantages, AMRSS also has limitations that point to important directions for further work. A key dependency is the choice of variability score $v(I)$. If $v(I)$ fails to align with

perceptual complexity or task-relevant structure, the algorithm may allocate too many samples to regions that appear visually uninteresting to users while undersampling segments that contain subtle but important features. Range-based scores are simple and effective for signals dominated by large excursions, but they may overemphasise isolated outliers; deviation-from-linearity scores capture shape better but are more expensive to compute and may still not fully reflect human judgements of salience. Similar trade-offs arise in other visualization-aware aggregation methods, where error metrics that are mathematically convenient do not always correlate with human performance on analytic tasks [1, 2]. Systematic exploration of variability scoring functions, possibly informed by perceptual models or learned from user studies, remains an open problem.

Another challenge concerns the robustness of thresholds and budget allocation strategies across datasets and application domains. Parameters such as the variability threshold τ, the relative weighting of $v(I)$ versus pixel span, and the mapping from pixel width to budget B may need to be tuned differently for seismic monitoring, physiological data analysis, and infrastructure telemetry. Prior work on min–max caching and visualization-aware time-series aggregation has shown that error bounds and caching policies can be derived analytically for certain metrics [5, 6], suggesting that analogous analyses for AMRSS could provide principled guidance on parameter selection. Developing data-driven or adaptive schemes that learn effective thresholds and prioritisation rules online, while maintaining theoretical guarantees on envelope coverage or visual error, is an appealing avenue for future research.

The current formulation of AMRSS also assumes a static or slowly changing time series and a finite time horizon. In high-throughput streaming scenarios and long-running monitoring deployments, maintaining and updating a deep hierarchical index raises questions about memory usage, tree rebalancing, and the treatment of very old data. While incremental update strategies and sliding-window variants are conceptually straightforward extensions of the framework, their performance and stability characteristics over long time horizons need to be carefully studied, drawing on ideas from streaming aggregation and online time-series indexing [2, 5, 6]. Similarly, extending AMRSS to multichannel settings, where cross-channel relationships and synchronisation constraints matter, will require joint variability measures and coordinated refinement policies that preserve correlations across channels.

Finally, there is substantial scope to integrate user tasks and perceptual factors more directly into the design of AMRSS. Different analytic goals emphasise different aspects of the signal: anomaly detection workflows care most about preserving rare spikes and abrupt changes; trend analysis prioritises the faithful rendering of low-frequency components; and diagnostic applications may value the precise shape of recurring motifs, such as ECG waves or seismic phases [1, 2]. Perceptually motivated variability scores, task-specific error metrics, and learning-based refinement policies trained on user performance data could all be incorporated within the AMRSS framework to better align sampling decisions with end-user objectives. Bridging the gap between the algorithmic formulation presented here and user-centred evaluation—through controlled experiments comparing AMRSS and baseline methods on concrete visual tasks—would not only validate the practical utility of the approach but also inform the broader design of visualization-aware sampling schemes [9]. In this sense, AMRSS is best viewed as both a concrete method and a flexible foundation on which more specialised, task-aware, and perceptually grounded time-series visualization systems can be built.

6. Conclusion

We have presented Adaptive Multi-Resolution Stratified Sampling (AMRSS), a method for interactive visualization of massive time series that combines stratified min–max sampling with a hierarchical summary tree and adaptive query-time refinement. By aligning sampling resolution with local variability and viewport constraints, AMRSS offers a principled way to respect strict sample budgets while preserving critical visual features. The framework naturally supports multi-resolution interaction patterns and can be extended to multivariate and streaming settings.

While our focus has been on method design and experimental protocol, the proposed approach opens several avenues for further research. These include deriving formal error bounds for adaptive hierarchical summaries, integrating perceptual and task-based objectives into variability scoring, and conducting user studies to assess the impact of sampling strategies on analytic performance. We hope that AMRSS can serve both as a practical tool for building time-series dashboards and as a conceptual foundation for subsequent work on scalable, perception-aware time-series visualization.

References

[1] Keim, D. A., Schneidewind, J., & Sips, M. (2006, April). Scalable pixel based visual data exploration. In *Visual Information Expert Workshop* (pp. 12-24). Berlin, Heidelberg: Springer Berlin Heidelberg.

[2] Ali, M., Alqahtani, A., Jones, M. W., & Xie, X. (2019). Clustering and classification for time series data in visual analytics: A survey. *IEEE Access, 7*, 181314-181338.

[3] Steinarsson, S. (2013). *Downsampling Time Series for Visual Representation* (Doctoral dissertation).

[4] Chun, J. (2021). SentimentArcs: A novel method for self-supervised sentiment analysis of time series shows SOTA transformers can struggle finding narrative arcs. arXiv preprint arXiv:2110.09454.

[5] Nizam, H., Zafar, S., Lv, Z., Wang, F., & Hu, X. (2022). Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT. *IEEE Sensors Journal, 22*(23), 22836-22849.

[6] Maroulis, S., Stamatopoulos, V., Papastefanatos, G., & Terrovitis, M. (2024). Visualization-aware Time Series Min-Max Caching with Error Bound Guarantees. *Proceedings of the Vldb Endowment, 17*(8), 2091-2103.

[7] Hao, M. C., Dayal, U., Keim, D. A., & Schreck, T. (2007). *Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data.* (Report).

[8] Madala, S. (2022). U.S. Patent No. 11,269,877. Washington, DC: U.S. Patent and Trademark Office.

[9] Wang, J., Zhao, Y., Chen, J., Zhang, S., Zhao, X., & He, Y. (2019). Efficient stratified sampling graphing method for mass data. *Data Science Journal, 18*, 56-56.

[10] Aigner, W., Miksch, S., Schumann, H., & Tominski, C. (2011). *Visualization of Time-Oriented Data* (Vol. 4). London: Springer.

[11] Aigner, W., Miksch, S., Müller, W., Schumann, H., & Tominski, C. (2007). Visualizing time-oriented data—a systematic view. *Computers & Graphics, 31*(3), 401-409.

[12] Heer, J., & Shneiderman, B. (2012). Interactive dynamics for visual analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue, 10*(2), 30-55.

[13] Shneiderman, B. (2003). The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization* (pp. 364-371). Morgan Kaufmann.

[14] Cleveland, W. S., & McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association, 79*(387), 531-554.

[15] Tufte, E. R., & Graves-Morris, P. R. (1983). *The Visual Display of Quantitative Information* (Vol. 2, No. 9). Cheshire, CT: Graphics press.

[16] Van Wijk, J. J., & Van Selow, E. R. (1999, October). Cluster and calendar based visualization of time series data. In *Proceedings 1999 IEEE Symposium on Information Visualization* (InfoVis' 99) (pp. 4-9). IEEE.

[17] McLachlan, P., Munzner, T., Koutsofios, E., & North, S. (2008, April). LiveRAC: interactive visual exploration of system management time-series data. In *Proceedings of the Sigchi Conference on Human Factors in Computing Systems* (pp. 1483-1492).

[18] Byron, L., & Wattenberg, M. (2008). Stacked graphs–geometry & aesthetics. *Ieee Transactions on Visualization and Computer Graphics, 14*(6), 1245-1252.

[19] Hochheiser, H., & Shneiderman, B. (2004). Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization, 3*(1), 1-18.

How to cite this article: William W. Predebon (2024). Adaptive Multi-Resolution Stratified Sampling for Interactive Visualization of Massive Time Series. *Bulletin of Computer and Data Sciences*, 5(4), 60-73. DOI: 10.71448/bcds2454-5

Received: 16/09/2024 **Revised:** 14/10/2024 **Accepted:** 27/11/2024 **Publish:** 30/12/2024

Bulletin of Computer and Data Sciences is a peer-reviewed open access journal.