DigitHist: a Histogram-Based Data Summary with Tight Error Bounds

Michael Shekelyan
Faculty of Computer Science
Free University of
Bozen-Bolzano, Italy
mshekelyan@unibz.it

Anton Dignös
Faculty of Computer Science
Free University of
Bozen-Bolzano, Italy
dignoes@inf.unibz.it

Johann Gamper
Faculty of Computer Science
Free University of
Bozen-Bolzano, Italy
gamper@inf.unibz.it

ABSTRACT

We propose DigitHist, a histogram summary for selectivity estimation on multi-dimensional data with tight error bounds. By combining multi-dimensional and one-dimensional histograms along regular grids of different resolutions, DigitHist provides an accurate and reliable histogram approach for multi-dimensional data. To achieve a compact summary, we use a sparse representation combined with a novel histogram compression technique that chooses a higher resolution in dense regions and a lower resolution elsewhere. For the construction of DigitHist, we propose a new error measure, termed u-error, which minimizes the width between the guaranteed upper and lower bounds of the selectivity estimate. The construction algorithm performs a single data scan and has linear time complexity. An in-depth experimental evaluation shows that DigitHist delivers superior precision and error bounds than state-of-the-art competitors at a comparable query time.

1. INTRODUCTION

Selectivity estimation based on data summary structures plays a crucial role for query optimization and approximate query answering in applications such as OLAP and decision support systems. A critical aspect of summary structures is reliability, i.e., how reliable is the query result derived from the structure. Histograms summarize data points by grouping them into buckets and counting the number of data points per bucket. The resulting aggregation of the data can be used to deduce guaranteed lower and upper bounds for the selectivity of range queries. This is not possible with non-histogram approaches, such as random sampling, kernel density estimation or wavelets. They only permit to obtain confidence intervals for query results that are probabilistic in nature and therefore less reliable than the bounds provided by histograms. Though the bounds are guaranteed, existing histogram approaches become imprecise when datasets are large and have more than a few dimensions. To tackle this problem, we propose *DigitHist*, a histogram that provides tight bounds and accurate selectivity estimates for large datasets and a moderate number of dimensions.

Consider the example in Figure 1. It shows a large spatial dataset of around 2.9 billion GPS coordinates around the world zoomed in on Italy (Figure 1a), which were collected for the OpenStreetMap project[1]. The query region of size 100 km around Rome contains 6.2 million points and has a (true) selectivity of $\frac{6.2 \cdot 10^6}{2.9 \cdot 10^9} = 0.214\%$, i.e., 0.214% of all data points are inside the query region. The figure shows two histogram summaries of the data, visualized as density heat maps depicting denser regions with darker shades. MinSkew [2], a state-of-the-art summary approach for spatial data, in Figure 1b estimates the selectivity of the query to be 0.177% with guaranteed bounds of $[0.102\%, 0.24\%]$; the relative estimation error is 17.3%. In contrast, the proposed DigitHist approach of roughly the same size in Figure 1c estimates the selectivity to be 0.215% with bounds $[0.195\%, 0.251\%]$; the relative estimation error is 0.5%. Thus, DigitHist has $2.4\times$ tighter bounds and a $34\times$ smaller estimation error than the MinSkew summary. Averaging over all 193 world capitals, DigitHist has $4.8\times$ tighter bounds and a $3.5\times$ smaller estimation error.

(a) Data (46.4GB) **(b)** MinSkew (113kB) **(c)** DigitHist (97kB)

	MinSkew	DigitHist
Estimated selectivity $sel_H(Q)$	0.177%	0.215%
Relative error	17.3%	0.5%
Bounds	$[0.102\%, 0.24\%]$	$[0.195\%, 0.251\%]$
Width of bounds $w_H(Q)$	0.138%	0.056%

(d) Estimates and bounds of MinSkew and DigitHist

Figure 1: Summary Structures: MinSkew vs. DigitHist.

A DigitHist summary is comprised of a small number of multi-dimensional equi-width histograms, termed *digit histograms*, each of which is augmented with a one-dimensional *marginal histogram* for each dimension; all histograms are along regular grids. The digit histograms summarize disjoint subsets of the data at different resolutions with a higher resolution for denser parts of the data. For instance, Figure 2 shows the four digit histograms of DigitHist (without the marginals) for the dataset in Figure 1 together with the percentage of data points summarized by each histogram, its resolution and size. The first two histograms summarize almost 90% of the data at a high resolution, consuming less than half of the summary size. The remaining data points are summarized at

[1] www.openstreetmap.org

Proceedings of the VLDB Endowment, Vol. 10, No. 11

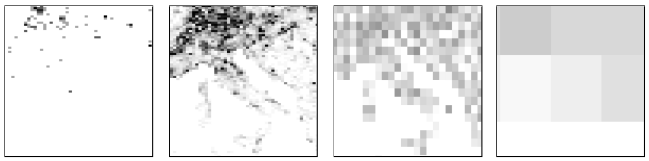


Figure 2: Digit histograms D_3, D_2, D_1 and D_0.

a lower resolution since they have a smaller impact on the histogram's precision. Unlike previous histogram approaches, such as equi-width and equi-depth, which group data points into buckets based on either location or local density, DigitHist groups data based on both location and local density. As a result, the summary has good knowledge about the location of most points.

As the bucket granularity necessarily decreases with an increasing dimensionality of the data for a given summary size, we augment each digit histogram with a one-dimensional marginal histogram at a higher resolution for each dimension. The marginal histograms together with the attribute value independence assumption are used to spread the data points inside the buckets of the digit histograms. This information significantly improves the precision when the query region partially overlaps buckets. Previous approaches assume a uniform data distribution inside buckets, which is far less accurate. The problem with only using one-dimensional histograms is that the attribute value independence assumption is typically violated in real-world data, resulting in low accuracy. Using both multi-dimensional and one-dimensional histograms allows us to correct the estimates of the attribute value independence assumption, such that the estimates never contradict the multi-dimensional histograms. We thereby combine the knowledge of the multi-dimensional histograms about the dependencies between attributes and the knowledge of the one-dimensional histograms about the distribution of attribute values.

The construction of DigitHist requires only one scan of the data. During this scan, the data is aggregated at a very high resolution into an initial multi-dimensional histogram and a one-dimensional histogram for each dimension. To achieve otherwise unattainable high resolutions for skewed data, only non-empty grid cells are materialized. The size of the initial histogram is then reduced to fit a desired summary size by using a novel lossy compression technique. The basic idea is to first decompose the initial histogram along the digits of the largest bucket count, i.e., one digit histogram for each digit position. For instance, a histogram with four buckets and respective counts $[10, 152, 1009, 12]$ would be decomposed into four digit histograms $D_3 = [0, 0, 1000, 0]$, $D_2 = [0, 100, 0, 0]$, $D_1 = [10, 50, 0, 10]$ and $D_0 = [0, 2, 9, 2]$, where D_3 has only multiples of thousand, D_2 only multiples of hundred, and so on. The digit histograms are then compressed by reducing their resolution while minimizing the information loss. This can be formulated as a multiple choice knapsack problem. As the digit histograms store significantly different amounts of information, their resolutions will differ, too. It is easy to see that the bucket counts of the digit histograms can be stored with small numbers and a multiplicative factor. In combination with storing only non-empty grid cells, we achieve a very compact representation of the proposed summary structure.

To measure the information loss during the lossy compression step, we propose a novel error metric, termed *u-error*, that measures a histogram's uncertainty about the data, expressed as the expected width of the histogram's selectivity bounds assuming a random distribution of range queries. The computation of the u-error is linear in the summary size and independent of the data size. Existing error metrics optimize for a uniform spread inside buckets from which we diverge to better deal with higher dimensional data. For instance, spatial skew [2] was proposed as a metric for histograms over real-valued data and later improved in [21]. The improved spatial skew metric discretizes the data space along a regular grid, pretends that each grid cell is a discrete value and computes the V-error [10] over values grouped into buckets. The V-error is defined as the sum of weighted variances $\sum_{j=1}^{J} n_j V_j$, where J is the number of buckets, n_j is the number of values and V_j is the variance of the value frequencies in bucket j. The V-error optimizes for small density variances inside buckets to reduce estimation errors when assuming a uniform spread of points inside buckets. In comparison, the u-error optimizes for tight bounds and also works well for more sophisticated spread assumptions.

The main technical contributions can be summarized as follows:

- We propose a new error measure, termed u-error, to measure a histogram's quality by calculating the expected width of the bounds for random range queries.
- We introduce DigitHist, a new summary structure over regular grids that is composed of a set of multi-dimensional histograms representing disjoint subsets of the data at different granularity levels, each of which is augmented by a one-dimensional histogram for each dimension.
- We propose a new histogram compression technique that chooses higher resolutions in dense regions and lower resolutions elsewhere, while minimizing the u-error and compressing the summary to a given size.
- Experiments on real-world data show that DigitHist provides more accurate selectivity estimates and tighter bounds than its competitors at a comparable query time.

The rest of the paper is organized as follows. In Section 2 related work is discussed. In Section 3, the basic concepts of multi-dimensional histograms and selectivity estimation are introduced. Section 4 presents the error measure u-error, which is used for the construction of DigitHist. In Section 5, the DigitHist summary is described in detail. Section 6 compares DigitHist to random sampling and popular histogram approaches.

2. RELATED WORK

There exists a large body of work on multi-dimensional selectivity estimation inspired by one-dimensional histograms, statistical methods and compression techniques. For a comprehensive overview of existing approaches we refer to [9, 6, 4].

Multi-dimensional selectivity estimation can be classified into two types of approaches: Data-driven approaches scan the data upfront to create the summary, whereas self-tuning approaches [1, 3, 8] use query answers of the database system to dynamically construct a summary. A key property of self-tuning approaches is that they organically prioritize frequently queried regions, but they cannot deduce bounds and are inaccurate when an insufficient number of previous queries intersects the current query region.

A popular data-driven approach are multi-dimensional histograms. A simple example are equi-depth histograms [14] that group the data points into buckets containing roughly the same number of points. They excessively prioritize denser regions of the data, while less dense regions are almost completely neglected. For large datasets, the construction requires multiple data scans to either externally sort the data multiple times or to compute multiple quantiles that depend on each other. Our DigitHist approach scans

the data only once and, while it also prioritizes denser regions of the data, it is not as radical as equi-depth histograms.

V-optimal histograms [10, 17, 11] received a lot of attention because they offer a clear notion of optimality for discrete-valued data. The construction for more than one dimension is known to be NP-hard. There are multiple ways to generalize it to real-valued data, but it is unclear if the general idea of minimizing density variances inside buckets is the best strategy in the multi-dimensional case. We show in this work that using one-dimensional histograms for the spread of points inside multi-dimensional buckets appears to offer vastly more accurate estimates. In addition, optimizing for density variances requires looking at the data, which is expensive for large datasets. We propose a novel error metric, u-error, that, instead of measuring density variances inside buckets, measures how much the histogram knows about the distribution of data points.

MinSkew [2] shares the basic idea of V-optimal histograms, but settles for sub-optimal histograms in order to avoid excessive construction costs. MinSkew first creates a histogram along a regular grid and then greedily merges grid cells that minimize the variance of grouped bucket counts until the desired number of buckets is reached. MHist-2 [16] is another greedy technique and in many aspects similar to MinSkew. The problem with these approaches is that they make split decisions using heuristics based on marginal distributions, which does not work well for skewed data.

GenHist [5] repeatedly moves high-density bumps in the data into buckets to flatten the density function. It inspired our approach to use overlapping buckets, sparsely populated grids and smaller bucket regions for denser data regions. Unlike other histogram approaches, GenHist maintains only approximate bucket counts and hence cannot deduce bounds. Moreover, it suffers from a very slow construction time. Our approach addresses these shortcomings.

The non-histogram approaches most related to our work are lossy compression techniques, such as DCT [12] and wavelet-based approaches [13, 19]. They start with an initial histogram and then lossy compress it by approximating the number of points counted by each bucket. In comparison, our approach produces histograms that allow to deduce guaranteed bounds. Another important non-histogram approach is random sampling [7]. It uses a random subset of the data to estimate the selectivity, which has the advantage that, with a fixed sample size, the estimation accuracy does not deteriorate with increasing dimensionality. Kernel density estimation [5] measures the density at different points in the data space to estimate the selectivity. It is expensive to construct and difficult to determine a good radius in which points should contribute to the density; a GPU-accelerated self-tuning approach [8] was proposed to remedy this problem. A problem of non-histogram approaches is that they cannot derive deterministic bounds, which limits their usefulness for some applications.

Another interesting line of research are ε-approximation summaries in computational geometry and related fields. They offer the same $\pm\varepsilon n$ error bounds for all datasets, where n is the number of data points. Quantile-based approaches create ε-approximations by splitting the data into quantiles akin to equi-depth histograms and borrow ideas from range trees to achieve faster querying and error guarantees. A notable approach is to use nested one-dimensional quantile summaries [18], which can be created in d data scans and guarantee a size of $\mathcal{O}(\frac{1}{\varepsilon}\log(\varepsilon n)\log^{2d-2}(\frac{1}{\varepsilon}\log(\varepsilon n))$ for a d-dimensional dataset with cardinality n. Another approach is a specialized two-dimensional technique [20], which achieves asymptotically optimal storage costs $\mathcal{O}(\frac{1}{\varepsilon}(\log^2\frac{1}{\varepsilon}+\log n)$ paired with $\mathcal{O}(\log\frac{1}{\varepsilon})$ query time, but requires to sort the data multiple times. Deterministic sampling approaches create ε-approximations by finding a representative subset of the data. The most notable approach is [15], which has $\mathcal{O}(\frac{1}{\varepsilon}\log^{2d}(\frac{1}{\varepsilon})\,\text{polylog}(\log(\frac{1}{\varepsilon})))$ storage costs, but requires a large construction time of $\mathcal{O}(n\frac{1}{\varepsilon^3}\text{polylog}(\frac{1}{\varepsilon}))$.

Major limitations of these ε-approximations are their comparably high storage and construction costs, as they do not exploit common data patterns to reduce storage costs, and both the storage and construction costs increase exponentially in the dimensionality. Thus, even for a relatively small number of dimensions they fail to provide tight error guarantees for a limited space budget. In comparison, the proposed DigitHist approach does not provide error guarantees for all queries, but typically offers tight bounds for individual queries even for a moderate number of dimensions and limited space budget.

3. PRELIMINARIES

Let $\mathcal{D} \subseteq [0,1]^d$ be a normalized d-dimensional dataset. The *selectivity*, $sel(Q)$, of a rectangular query region $Q \in \mathbb{R}^d \times \mathbb{R}^d$ over $\mathcal{D}$ is defined as the fraction of data points that are contained in Q, i.e., $sel(Q) = \frac{|\{p|p\in\mathcal{D}\wedge p\in Q\}|}{|\mathcal{D}|}$. A d-dimensional *histogram*, H, with J buckets is comprised of a set of bucket regions, $r_1^H,\ldots,r_J^H$, with corresponding bucket counts, $f_1^H,\ldots,f_J^H$, where each bucket region $r_j^H \in \mathbb{R}^d\times\mathbb{R}^d$ is a d-dimensional hyper-rectangle and each bucket count $f_j^H \in \mathbb{N}_0$ is a non-negative integer. Histograms provide bounds for the true selectivity $sel(Q)$ of queries. The *lower bound*, $sel_H^{LB}(Q)$, is computed by adding the counts of all buckets that are contained in the query region and dividing the result by the data size. The *upper bound*, $sel_H^{UB}(Q)$, is computed by adding the counts of all buckets that are intersected by the query region and divide the result by the data size. We refer to the gap between the lower and upper bound as the *width* of the bounds for a query Q, which is defined as $w_H(Q) = sel_H^{UB}(Q) - sel_H^{LB}(Q)$. For instance, consider a histogram that summarizes 1000 points and a query Q that completely covers two buckets with 60 and 40 points, respectively, and partially covers two other buckets with 28 and 2 points, respectively. The $60+40$ points are certain to be in the query region, whereas the other $28+2$ points are not. This yields a selectivity lower bound of $\frac{60+40}{1000} = 0.1$, an upper bound of $\frac{60+40+28+2}{1000} = 0.13$ and a width of bounds $w_H(Q) = \frac{28+2}{1000} = 0.03$. Table 1 summarizes the most important notation used in the paper.

Table 1: Notation Table

Notation	Description
$\mathcal{D}$	set of data points
d	number of dimensions of data points
H	histogram
f_j^H	number of points counted by bucket j in H
r_j^H	bucket region of bucket j in H
$sel_H(Q)$	selectivity estimate of histogram H for Q
$sel_H^{LB}(Q)$ / $sel_H^{UB}(Q)$	H's lower/upper bound for selectivity of Q
$w_H(Q)$	width of H's bounds for Q
D_k	digit histogram for the k-th digit
M_k^i	marginal histogram in dimension i for D_k

4. OPTIMIZATION MEASURE: u-ERROR

In this section, we present the u-error, a new optimization measure for histogram construction. Instead of optimizing for how points are spread inside buckets, we minimize a histogram's uncertainty about the data, i.e., the expected number of points about

which the histogram does not know if they are inside the query region or not.

4.1 Definition of the u-Error

We assume a set $\mathcal{Q}$ of random hyper-cube queries that are uniformly distributed and completely contained in the normalized data space. The following definition specifies the query distribution $\mathcal{C}^d$ by describing how to generate random queries.

Definition 1. (*Random query region*) A *random query region* $X \sim \mathcal{C}^d$ is drawn by first drawing its volume v uniformly from the range $[0, 1]$ and then drawing its center coordinates p_i uniformly from the range $[\frac{\sqrt[d]{v}}{2}, 1 - \frac{\sqrt[d]{v}}{2}]$ for each dimension i.

Definition 2. (*u-error*) Let $X \sim \mathcal{C}^d$ be a random query and $Pr[X = Q]$ be the probability that X is the query region $Q \in \mathcal{Q}$. The *u-error* of a histogram H is defined as

$$uerr(H) = E[w_H(X)] = \sum_{Q \in \mathcal{Q}} Pr[X = Q] w_H(X).$$

The u-error has a histogram H as an input and outputs the expected width of H's bounds for a random query region. The u-error has three important properties: it takes values between 0 % and 100 %; the computational complexity is linear in the histogram size and independent of the data size; and no assumption is made about how points are spread inside buckets.

4.2 Computing the u-Error

The following theorem provides an effective way to compute the u-error.

THEOREM 1 (u-ERROR COMPUTATION). *Let H be a histogram with J buckets and $X \sim \mathcal{C}^d$ be a random variable representing a query region. The u-error of H is computed as*

$$uerr(H) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{J} Pr[X \text{ overlaps } r_j^H] f_j^H,$$

where $Pr[X \text{ overlaps } r_j^H]$ is the probability that a query drawn from the random distribution overlaps the bucket region r_j^H.

PROOF. Let $I[S]$ be equal to 1 if the statement S is true and equal to 0 if S is false, and let $w_H^{(j)}(Q) = I[Q \text{ overlaps } r_j^H] f_j^H$. Then, we have:

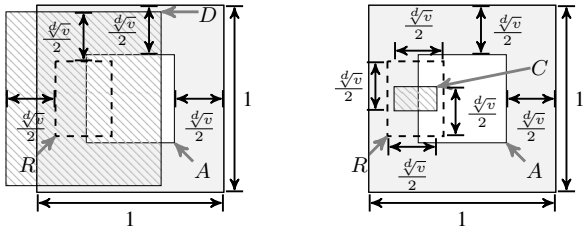
$$\begin{aligned}
&\frac{1}{|\mathcal{D}|} \sum_{j=1}^{J} Pr[X \text{ overlaps } r_j^H] f_j^H \\
&\quad = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{J} E[\, I[X \text{ overlaps } r_j^H] f_j^H \,] \\
&\quad = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{J} \sum_{Q \in \mathcal{Q}} Pr[Q] \left(I[Q \text{ overlaps } r_j^H] f_j^H \right) \\
&\quad = \sum_{Q \in \mathcal{Q}} \left[Pr[Q] \left(\frac{1}{|\mathcal{D}|} \sum_{j=1}^{J} I[Q \text{ overlaps } r_j^H] f_j^H \right) \right] \\
&\quad = E[w_H(X)] = uerr(H) \quad \square
\end{aligned}$$

The theorem shows that the u-error can be computed by summing the contribution $Pr[X \text{ overlaps } r_j^H]$ of each bucket. To compute these contributions, we use the following two lemmas and the equivalence $Pr[X \text{ overlaps } r_j^H] = Pr[X \text{ intersects } r_j^H] - Pr[X \text{ contains } r_j^H]$, where ($A$ intersects B) $\Leftrightarrow$ ($(A \cap B) \neq \varnothing$) and ($A$ contains B) $\Leftrightarrow$ ($(A \cap B) = B$). The integrals in the following lemmas can be numerically integrated in $\mathcal{O}(d)$. Since there are J buckets, the overall time complexity of computing the u-error is $\mathcal{O}(dJ)$. The time complexity is independent of the data size since only the histogram is accessed and not the data.

LEMMA 1 (INTERSECTION PROBABILITY). *Let $X \sim \mathcal{C}^d$ be a hyper-cube, R be a hyper-rectangle, $[a_i, b_i]$ be the range of R in dimension i and $s_i(v) = \min\{b_i + \frac{\sqrt[d]{v}}{2}, 1 - \frac{\sqrt[d]{v}}{2}\} - \max\{a_i - \frac{\sqrt[d]{v}}{2}, \frac{\sqrt[d]{v}}{2}\}$. The intersection probability of X and R is computed as*

$$Pr[X \textit{ intersects } R] = \int_0^1 \frac{1}{1 - \sqrt[d]{v}} \prod_{i=1}^{d} s_i(v)\, dv.$$

PROOF. Let X be a hyper-cube with volume v and D be the region where X's center would have to lie such that X intersects R (hatched rectangle in Fig. 3a). The interval of D in dimension i is then equal to $[a_i - \frac{\sqrt[d]{v}}{2}, b_i + \frac{\sqrt[d]{v}}{2}]$. Furthermore, let A be the region where X's center would have to lie such that X is completely contained in the data space (white rectangle in Fig. 3a). The interval of V in dimension i is equal to $[\frac{\sqrt[d]{v}}{2}, 1 - \frac{\sqrt[d]{v}}{2}]$. Since X's center is uniformly distributed in A, the probability that a random square intersects R is $\frac{v(D \cap A)}{v(A)} = \frac{\prod_{i=1}^{d} s_i}{1 - \sqrt[d]{v}}$. By integrating over all possible hyper-cube volumes we obtain $Pr[X \text{ intersects } R]$. $\square$



(a) Case X intersects R **(b)** Case X contains R

Figure 3: Illustration of intersection and contain probabilities.

LEMMA 2 (CONTAIN PROBABILITY). *Let $X \sim \mathcal{C}^d$ be a hyper-cube, R be a hyper-rectangle, $[a_i, b_i]$ be the range of R in dimension i, $t_i(v) = \min\{a_i + \frac{\sqrt[d]{v}}{2}, 1 - \frac{\sqrt[d]{v}}{2}\} - \max\{b_i - \frac{\sqrt[d]{v}}{2}, \frac{\sqrt[d]{v}}{2}\}$ and $m = \max_{1 \le i \le d} b_i - a_i$. The probability that X contains R is computed as*

$$Pr[X \textit{ contains } R] = \int_m^1 \frac{1}{1 - \sqrt[d]{v}} \prod_{i=1}^{d} t_i\, dv.$$

PROOF. Let X be a hyper-cube with volume v and C be the region where X's center would have to lie such that X contains R (hatched rectangle in Fig. 3b). The interval of C in dimension i is then equal to $[b_i - \frac{\sqrt[d]{v}}{2}, a_i + \frac{\sqrt[d]{v}}{2}]$. Furthermore, let A be the region where X's center would have to lie such that X is completely contained in the data space (white rectangle in Fig. 3b). The interval of A in dimension i is equal to $[\frac{\sqrt[d]{v}}{2}, 1 - \frac{\sqrt[d]{v}}{2}]$. Since X's center is uniformly distributed in A, the probability that a random square contains R is therefore $\frac{v(C \cap A)}{v(A)} = \frac{\prod_{i=1}^{d} t_i}{1 - \sqrt[d]{v}}$. By integrating over all possible hyper-cube volumes we obtain $Pr[X \text{ contains } R]$. If $v < m = \max_{1 \le i \le d} b_i - a_i$, then $Pr[X \text{ contains } R] = 0$, because X is too small to contain R. It is therefore sufficient to integrate from m to 1. $\square$

5. DATA SUMMARY: DigitHist

5.1 Constructing DigitHist

The basic idea of DigitHist is to summarize the data by a set of equi-width multi-dimensional and one-dimensional histograms with different resolutions.

Definition 3. (DigitHist) The *DigitHist* summary of a d-dimensional dataset $\mathcal{D}$ is composed of a set of d-dimensional equi-width histograms $D_0, \ldots, D_{K-1}$, termed *digit histograms*, and for each D_k a set of d one-dimensional equi-width histograms $M_k^1, \ldots, M_k^d$, termed *marginal histograms*.

Each digit histogram D_k summarizes a disjoint subset of the data using a different grid resolution. Histograms that summarize more points typically use a higher resolution. The one-dimensional histograms, M_k^d, associated to the digit histograms are used to spread the data points inside the buckets of the digit histograms. The construction of the DigitHist summary proceeds in three steps:

1. Create an initial d-dimensional histogram H and, for each dimension i, a one-dimensional marginal histogram M^i.
2. Decompose and compress H into K digit histograms $D_{K-1}, \ldots, D_0$ of maximum S bytes.
3. Distribute each initial marginal histogram M^i over the marginal histograms $M_{K-1}^i, \ldots, M_0^i$ associated with $D_{K-1}, \ldots, D_0$.

5.1.1 Initial Histograms

The first step scans the data and constructs an initial multi-dimensional histogram and one marginal histogram per dimension.

Definition 4. (*Initial histograms*) Let $\mathcal{D}$ be a d-dimensional dataset. The *initial multi-dimensional histogram*, H, is an equi-width histogram that has the highest possible resolution such that all data points are located in at most B buckets and the grid resolution in each dimension is a power of two. The *initial marginal histograms*, $M^1, \ldots, M^d$, are one-dimensional equi-width histograms of a given resolution C, each summarizing the data projected onto one of the dimensions.

To achieve a high resolution of the initial multi-dimensional histogram, only non-empty buckets are materialized by storing them in a hash table indexed by the bucket address. The data scan begins with a very high resolution grid (e.g., 2^{62} buckets in our implementation, which is the largest power of two that can be represented with a signed 64-bit integer). When the number of non-empty buckets exceeds B, the resolution along one or more dimensions is reduced by merging pairs of adjacent buckets. The dimensions for merging are chosen in a round-robin fashion to prevent very stretched grid cells. This strategy ensures the construction of a multi-dimensional histogram with the highest possible precision we can support. Especially for skewed datasets, we can operate in orders of magnitude higher resolutions than otherwise would be possible with a dense representation. For the initial marginal histograms, we assume a fixed granularity C, which is a power of two.

Example 1. Figure 4 illustrates the construction of the initial histograms for a dataset of 2000 points. Only non-zero bucket counts are depicted. The maximal number of non-empty cells is $B = 15$, and the resolution of the initial marginal histograms $C = 8$. The initial grid has $2^{62} = 2^{31} \times 2^{31}$ cells. After processing 4.3% of the data points, the grid has already only 8×8 cells. Since there are more than 15 non-empty cells, the grid resolution has to be further reduced by merging adjacent cells yielding 4×8 cells. Processing the remaining data points yields a 4×4 grid, which is the highest resolution with at most 15 non-empty cells.

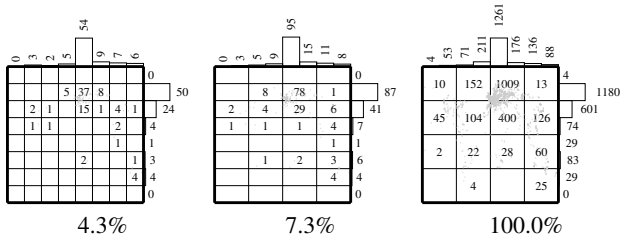


Figure 4: Initial histograms after processing % of the data.

5.1.2 Digit Histogram Compression

The digit histogram compression is comprised of a decomposition and a shrinking step. The decomposition splits the initial multi-dimensional histogram, H, into multiple digit histograms, $\hat{D}_{K-1}, \ldots, \hat{D}_0$, such that reducing the size of some digit histograms leads to a larger error than others. The subsequent shrinking step determines the optimal size reduction of the digit histograms, $D_{K-1}^*, \ldots, D_0^*$, such that they fit into a user-specified size constraint S and the total u-error is minimized. To get the best lossy compression of the initial histogram, the decomposition and shrinking steps are repeated with lower resolutions of the initial histogram as a starting point. The reason for trying lower resolutions of the initial histogram is that the decomposition operates on bucket counts and is most effective if they have a large disparity; at very high resolution all bucket counts tend to be very small.

Definition 5. (*Digit histograms*) Let H be an initial histogram and b be the radix of a numeral system such that all bucket counts in H can be represented by K digits, i.e., $f_j^H = x_{K-1} \ldots x_0$. The K *digit histograms* of H are defined as $\hat{D}_{K-1}, \ldots, \hat{D}_0$, where each $\hat{D}_k$ has the same resolution as H and the bucket counts are $f_j^{\hat{D}_k} = x_k \cdot b^k$.

The digit histogram decomposition transforms an initial multi-dimensional histogram into a set of digit histograms. The digit histograms summarize disjoint subsets of the data that originate from different digit positions. Since all bucket counts of a digit histogram $\hat{D}_k$ are multiples of the coefficient b^k, we store b^k only once and, for each bucket, the digit x_k, yielding a compact representation. In our implementation, we limit ourselves to numeral systems with a power of two as a basis b since digit-related operations can then be efficiently done as bit operations. For illustration purposes we use in the running example the decimal system.

Example 2. Figure 5 shows the decomposition of the initial histogram into $K = 4$ digit histograms. All digit histograms have the same bucket regions as the initial histogram, and they store only the corresponding digits of the bucket counts; the corresponding coefficient is shown in parentheses. For instance, the digit in the top-right bucket is a three in $\hat{D}_0$, a one in $\hat{D}_1$ and a zero in the other two digit histograms. Summing up $0 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 3 \cdot 10^0$ yields the top-right bucket count 13 of the initial histogram.

The digit histograms facilitate size reduction with small information loss because different resolutions can be used depending on the number of data points that are represented by a digit histogram.

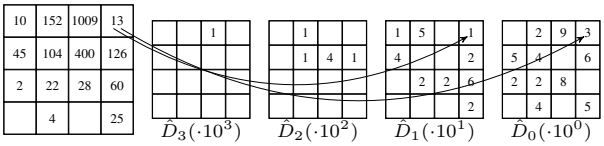


Figure 5: Decomposition into four digit histograms.

Definition 6. (*Resolution space*) The *resolution space* $res(\hat{D}_k)$ of a digit histogram $\hat{D}_k$ with 2^L buckets is the sequence of histograms $\langle \hat{D}_k^L, \ldots, \hat{D}_k^0 \rangle$ such that for any $l \geq 1$, the histogram $\hat{D}_k^{l-1}$ with 2^{l-1} buckets is obtained from $\hat{D}_k^l$ with 2^l buckets by merging pairs of adjacent buckets along one dimension.

The resolution space of an initial digit histogram is comprised of all its lower-resolution versions down to a histogram with a single bucket covering the entire data space. In each compression step from a resolution level l to $l-1$ information is lost, which is measured by using the u-error. At the same time, the number of non-empty buckets, and hence the space requirements of the histogram, is decreasing. In Fig. 6, each column shows the resolution space of a digit histogram in the running example, together with the induced u-error and the size of the histogram in bytes. Obviously, a lower resolution yields a larger error and a smaller size of the histogram.

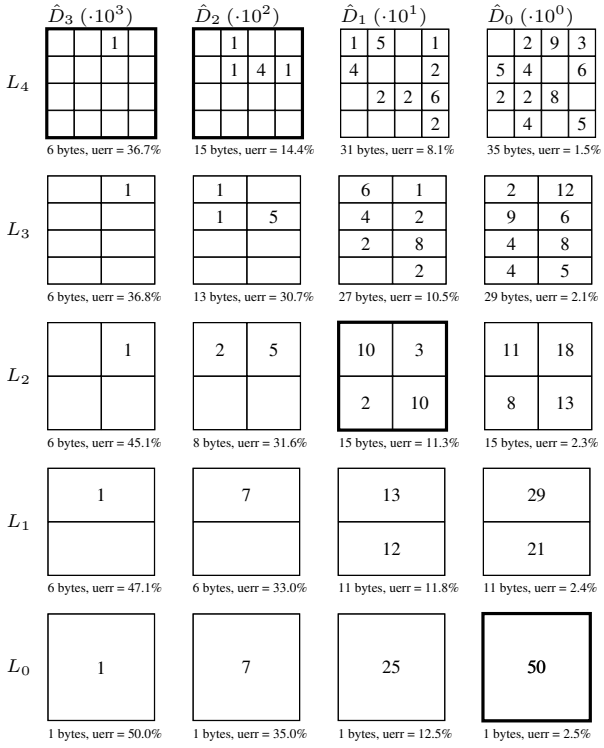


Figure 6: Resolution space with u-error and size.

In order to reduce the digit histograms to a user-defined size S, they are compressed while minimizing the induced error.

Definition 7. (*Optimal digit histogram resolutions*) Let S be the maximal number of bytes available for the digit histograms and $size(H)$ be the number of bytes needed to represent a histogram H. The *optimal set of digit histograms* $\langle D_{K-1}^*, \ldots, D_0^* \rangle$ with $D_k^* \in res(\hat{D}_k)$ minimizes $\sum_{k=0}^{K-1} uerr(D_k^*)$ under the constraint $\sum_{k=0}^{K-1} size(D_k^*) \leq S$.

The set of optimal digit histograms is comprised of one histogram from each digit histogram's resolution space such that the total number of bytes does not exceed the space budget and the total u-error is minimized.

The problem of finding the optimal resolution for each digit histogram can be formulated as a multiple choice knapsack problem. The multiple-choice knapsack solver gets as an input the size constraint S, the byte sizes $size(\hat{D}_k^L), \ldots, size(\hat{D}_k^0)$ and the negated errors $-uerr(\hat{D}_k^L), \ldots, -uerr(\hat{D}_k^0)$ for each digit histogram. It returns the optimal resolutions for each digit histogram such that the size constraint is satisfied and the error is minimized. Although the multiple-choice knapsack problem is weakly NP-hard, our problem instances are very small, and there exist good solvers for this purpose. In our implementation, we use $K = 4$ digit histograms and at most $G = 62$ resolution levels. Hence, we solve it in a brute-force fashion by considering all $G^K \approx 14 \cdot 10^6$ combinations, which takes less than 20 milliseconds.

Example 3. The four digit histograms at level L_4 in Fig. 6 have a total size of $6 + 15 + 31 + 35 = 87$ bytes. Assuming a space budget of at most 40 bytes for the digit histograms, the resolutions picked by the multiple-choice knapsack solver are highlighted in boldface. The two most significant digit histograms keep their initial resolution, whereas the other two histograms are compressed to a lower resolution. The selected combination of digit histograms induces an u-error of $36.7\% + 14.4\% + 11.3\% + 2.5\% = 64.9\%$ and requires 37 bytes.

The digit histogram compression algorithm is outlined in Algorithm 1. The input is the initial histogram, the summary size available to the digit histograms and the number of digit histograms. The output of the algorithm are the digit histograms, which are a lossy compression of the initial histogram. In line 1, the result set is initialized to empty histograms. In line 2, the termination condition is checked. The algorithm terminates when the initial histogram becomes small enough to obviate the need for compression. In line 3, the initial histogram is decomposed into K digit histograms (cf. Definition 5). In line 4 and 5, the optimal amount of shrinking for each digit histogram is determined (cf. Definition 7) by calling a multiple-choice knapsack solver. In line 6 and 7, the so far best found digit histograms are updated. Finally, line 8 reduces the size of the initial histogram for the next iteration.

Algorithm 1: DigitHistCompress

```
   Input: initial histogram H, summary size S, number of digit histograms K
1  ⟨D_{K-1}, ..., D_0⟩ ← ⟨∅, ..., ∅⟩ and let uerr(∅) = ∞
2  while size(H) > S do
3      decompose H into ⟨D̂_{K-1}, ..., D̂_0⟩
4      find combination ⟨D*_{K-1}, ..., D*_0⟩ with D*_k ∈ res(D̂_k)
5          s.t. Σ_{k=0}^{K-1} uerr(D*_k) is minimized and Σ_{k=0}^{K-1} size(D*_k) ≤ S
6      if Σ_{k=0}^{K-1} uerr(D*_k) < Σ_{k=0}^{K-1} uerr(D_k) then
7          ⟨D_{K-1}, ..., D_0⟩ ← ⟨D*_{K-1}, ..., D*_0⟩
8      reduce the resolution of H's grid by merging adjacent buckets along one
       dimension
9  return ⟨D_{K-1}, ..., D_0⟩
```

In order to improve the efficiency of this compression step, we only materialize digit histograms that are potentially small enough to fit into the final summary.

5.1.3 Marginal Histograms

The last step distributes the counts from each initial marginal histogram M^i to the digit histograms' marginals $\langle M_{K-1}^i, \ldots, M_0^i \rangle$ in

dimension i. We start by transferring counts to the marginal of the digit histogram with the highest resolution; if multiple digit histograms have the same grid resolution, a higher digit coefficient has precedence. For a multi-dimensional bucket j of a digit histogram D_k with bucket frequency $f_j^{D_k}$, we first determine which marginal buckets are covered by bucket j. Then, $f_j^{D_k}$ data points are moved from those buckets in the initial marginal histogram to the corresponding buckets of D_k's marginal histogram M_k^i. As the resolution of the marginal histograms is higher than the resolution of the digit histograms, we have to decide how many points we take from each marginal bucket. We simply take proportionally, i.e., if one marginal bucket has a twice as high count, we take twice as many from it. This procedure is repeated for all dimensions and all digit histograms. At the end, all data points from the initial marginal histograms are distributed to the marginals of the digit histograms.

Example 4. Consider the initial marginal histogram in the first (horizontal) dimension, $M^1 = \langle 4, 53, 71, 211, 1261, 176, 136, 88\rangle$, in Figure 4. Each digit histogram takes proportionally from this initial histogram without replacement. We start with D_3 and go through each bucket. The first (and only) bucket has count $1 \cdot 10^3 = 1000$. Hence, we take 1000 proportionally from the intersected marginal buckets with counts 1261 and 176, i.e., $\frac{1261}{1261+176}1000$ and $\frac{176}{1261+176}1000$. In order to avoid non-integral counts, we use a trivial greedy approach to round the numbers such that the sum is preserved and the absolute rounding error is minimized. This results in $M_3^1 = [0, 0, 0, 0, 878, 122, 0, 0]$. We subtract those counts from the initial marginal histogram, such that the initial marginal histogram has $[4, 53, 71, 211, 383, 54, 136, 88]$ left. Then we continue with the digit histogram D_2, which takes $[0, 0, 50, 150, 351, 49, 61, 39]$ and leaves $[4, 53, 21, 61, 32, 5, 75, 49]$ in the initial histogram. The procedure is repeated for D_1 and D_0. For D_0 the step becomes trivial, since it takes the remaining counts in the initial marginal histogram. The final DigitHist summary structure of our running example is shown in Figure 7.

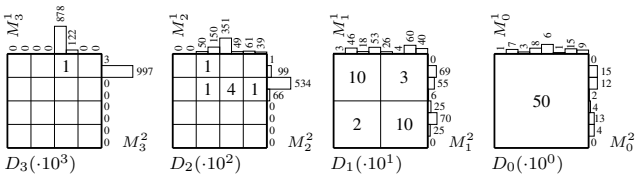
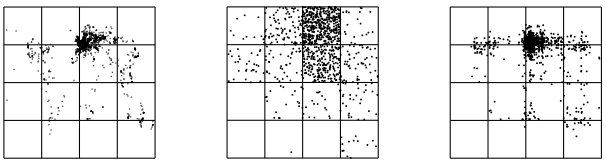


Figure 7: Final DigitHist summary.

5.2 Querying DigitHist

Querying DigitHist is similar to querying other histograms, except for how we deal with partially intersected buckets and that we have to query each of the K digit histograms. If the query region partially intersects a bucket, we estimate how many of the bucket's points are in the intersection. For this, we consult the marginal histograms to factor in the intersection's location. We call our approach *AVI spread* because it makes use of the attribute value independence (AVI) assumption to spread the points of a bucket over its bucket region. This is different from the classical AVI assumption, which spreads all points over the complete data space. Instead, we make an AVI assumption that is corrected by multi-dimensional information. This typically leads to a quite accurate picture of the data as illustrated in Figure 8, where it is compared to the uniformity assumption.



(a) data **(b)** uniform spread **(c)** AVI spread

Figure 8: Spread estimation inside buckets.

Definition 8. (DigitHist selectivity estimate) Let $\mathcal{D}$ be a d-dimensional dataset, Q a query range, D_k the digit histogram for the k-th digit, M_k^i the marginal histogram of dimension i for the k-th digit and π_i a function that projects a multi-dimensional range to the one-dimensional range of dimension i. The *DigitHist selectivity estimate* is calculated as follows:

$$sel_{DH}(Q) = \frac{1}{|\mathcal{D}|}\sum_{k=0}^{K-1}\sum_j f_j^{D_k}\alpha_j^{D_k}(Q), \text{ where}$$

$$\alpha_j^{D_k}(Q) = \begin{cases} 0 & \text{for } (Q \cap r_j^{D_k}) = \varnothing \\ \prod_{i=1}^{d} \frac{sel_{M_k^i}(\pi_i[Q \cap r_j^{D_k}])}{sel_{M_k^i}(\pi_i[r_j^{D_k}])} & \text{for } (Q \cap r_j^{D_k}) \subset r_j^{D_k} \\ 1 & \text{for } (Q \cap r_j^{D_k}) = r_j^{D_k} \end{cases}$$

and the selectivity of a marginal histogram M_k^i is calculated as $sel_{M_k^i}([a,b]) = \sum_{j=1}^{J} f_j^{M_k^i} \frac{d([a,b] \cap r_j^{M_k^i})}{d(r_j^{M_k^i})}$, where $d([a,b]) = b - a$.

The selectivity estimate is computed by going through all digit histograms and adding the counts of all buckets intersected by the query region. The counts of partially intersected buckets are multiplied by a factor, which estimates the fraction of the bucket's points that are inside the intersection with the query region. This factor divides the AVI estimate for the intersection by the AVI estimate for the bucket region, where an AVI estimate is simply the product of the one-dimensional selectivity estimates based on the marginal histograms.

The lower and upper bounds are computed by reusing the formula for the selectivity estimate. For the lower bound, one assumes that the data points of partially intersected buckets are not contained in the query region, and for the upper bound that they are contained in the query region. Additionally, the marginal histograms also provide an upper bound for the query region, which is only used if it is better.

Example 5. Figure 9 shows the final DigitHist summary with an exemplary query region (hatched rectangle). The selectivity lower bound of the query is equal to 0% as no bucket is completely covered by the query region. The selectivity upper bound is computed as $\frac{1}{2000}\left(10^3(1) + 10^2(4+1) + 10^1(1+2) + 10^0(50)\right) = 79\%$. The selectivity estimate of the query is computed as

$$\begin{aligned} sel_{DH}(Q) = {} & \frac{1}{2000}10^3\left[1\frac{122}{878+122}\frac{997}{997+3}\right] + \\ & \frac{1}{2000}10^2\left[4\frac{49}{49+351}\frac{534}{534+66} + 1\frac{61}{61+39}\frac{534}{534+66}\right] + \\ & \frac{1}{2000}10^1\left[3\frac{4+60}{26+4+60+40}\frac{69+55}{1+69+55+6}\right] + \\ & \frac{1}{2000}10^0\left[50\frac{1+15}{50}\frac{15+12}{50}\right] \approx 11.7\%. \end{aligned}$$

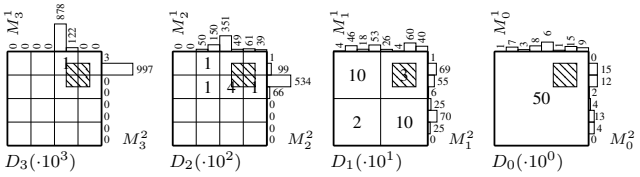


Figure 9: DigitHist summary and example query region.

5.3 Implementation Details

DigitHist is comprised of a set of multi-dimensional and a set of one-dimensional histograms, which have to be stored in a way that is both compact and can be efficiently queried.

For the representation of the one-dimensional marginal histograms, we use a dense representation and store cumulative sums instead of the bucket counts. Each range query can then be answered in $\mathcal{O}(1)$ time by looking up two cumulative sums.

For the multi-dimensional histograms, we use a sparse representation and store only the frequency of non-empty buckets. To address the buckets with an integral index, a z-order index is used as illustrated in Fig. 10. This index allows to efficiently access a cell's multi-dimensional coordinates as the index's bit representation is the interleaving of the coordinates' bit representations. For instance, the grid cell with index 9 in Fig. 10a has the bit string $[\underline{1}\hat{0}\underline{0}\hat{1}]_2$, where the hatted bits represent the coordinate in the first dimension and the underlined bits in the second dimension. The z-order index also facilitates the handling of multi-resolution grid schemes, which occur during the histogram construction. For instance, in Fig. 10a the least significant bit belongs to the first dimension. Removing the first bit from all indices and merging cells with equivalent indices results in the grid of Fig. 10b, which is equivalent to merging adjacent buckets along the first dimension. Since we compute during the construction of DigitHist the z-order index of each data point, we use in our implementation widely known look-up table techniques to speed-up the interleaving of bit strings.

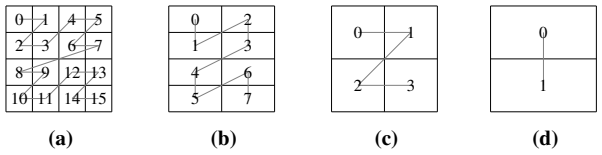


Figure 10: Z-order indices for different grid resolutions.

A sparse representation requires to store the bucket indices together with the bucket counts. In order to avoid representing the cell indices directly and to use an offset instead, we store the size of the gaps between non-empty buckets combined with the counts of the non-empty buckets, i.e., pairs of gap size and bucket count. The advantage of this gap representation is that it requires much less space for sparse histograms, and the overhead of reconstructing the cell indices at query time is minimal. To store the actual gap size and bucket counts, we use variable byte encoding, which uses 7 bits for the payload and 1 bit to indicate that the next byte is a continuation of the current payload.

In order to answer a query, one could naively go through all buckets, but the number of buckets can be large. To speed-up querying, we supplement each digit histogram with a lower resolution version that has a much smaller number of buckets. We then have to go only through all completely covered buckets of the less-detailed version and access the full-detail level only if a cell in the less-detailed version is partially intersected by the query region.

We call the cells of the lower resolution histogram *root cells*. Each root cell is comprised of a set of child cells, and its count sums the counts of its children. Figure 11 illustrates the digit histogram D_2 and the corresponding lower resolution histogram. For instance, the root cell with count 5 is comprised of the two child cells with 4 and 1. In our implementation, we pick for a digit histogram with N buckets a root cell grid resolution of $\sqrt{N}$ root cells.

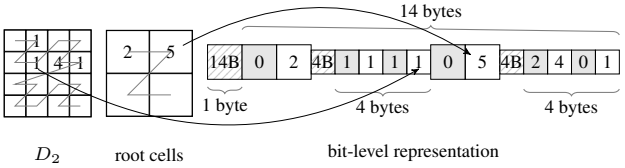


Figure 11: Representation of a digit histogram.

In the bit-level representation, the original digit histogram and the lower resolution histogram are intertwined so that we can skip the children of root cells that are completely covered by the query region. Figure 11 illustrates the bit-level representation of D_2. The first value is the number of bytes for the whole histogram, which is needed to determine when all root cells are read or to skip the whole histogram in the byte stream. Then the three values of the root cell follow: the gap value 0, the count value 2 and the children bytes value 4. These three values are followed by gap and count values of the first child and then the gap and count values of the second child. In order to skip the child cells, we simply have to skip the amount of children bytes, which is in this case 4.

5.4 Complexity Analysis

THEOREM 2 (CONSTRUCTION TIME). *DigitHist is constructed in $\mathcal{O}\big(d \cdot (|\mathcal{D}| + B + S)\big)$ time, where d is the data dimensionality, $|\mathcal{D}|$ is the number of data points, B is the maximal number of sparse buckets and S is the summary size.*

PROOF. Let G be the number of bits used for z-order indices to represent grid cells, which is a small constant. Let K be the number of digit histograms, which is also a small constant and independent of the data since the digit decomposition uses a numeral system such that all bucket counts have at most K digits. In the initial phase (cf. Section 5.1.1) for each of the $|\mathcal{D}|$ data points the z-order index is computed in $\mathcal{O}(d)$ and inserted/updated in the hash map in $\mathcal{O}(1)$. When we reach more than B sparse buckets, $\mathcal{O}(B)$ grid cell indices are divided by two (i.e., merged along one dimension), each in $\mathcal{O}(1)$ time. This may happen $G = \mathcal{O}(1)$ times and gives $\mathcal{O}(d \cdot |\mathcal{D}| + B)$ for the initial histogram. The lossy compression phase (cf. Section 5.1.2) consists of at most $G = \mathcal{O}(1)$ iterations. In each iteration the initial histogram at the current resolution is split into K digit histograms in $\mathcal{O}(B)$ operations. Then each of the K digit histograms is reduced in resolution at most $G = \mathcal{O}(1)$ times. For each of the at most $K \cdot G = \mathcal{O}(1)$ resulting histograms, the u-error and byte size of the histogram is computed in $\mathcal{O}(dS)$ operations. At the end of each iteration the multiple-choice-knapsack problem is solved in $G^K = \mathcal{O}(1)$. Hence, each iteration has time complexity $\mathcal{O}(B + S)$.

At the end of the lossy compression phase, each the initial d marginal histograms is divided up into K marginal histograms in $\mathcal{O}(dS)$ (cf. Section 5.1.3). Since there is a constant number of iterations, the lossy compression phase has worst case time complexity $\mathcal{O}(dS + dB)$. All phases together have time complexity $\mathcal{O}(d(|\mathcal{D}| + B + S)) + \mathcal{O}(dB + dS) = \mathcal{O}(d(|\mathcal{D}| + B + S))$. □

THEOREM 3 (INTERMEDIATE MEMORY). *The construction of DigitHist has worst case memory complexity $\mathcal{O}(dC + B + S)$, where d is the data dimensionality, C is the resolution of the marginal histograms, B is the maximal number of sparse buckets and S is the summary size.*

PROOF. During the initial phase, one multi-dimensional histogram with B values and d marginal histograms with C values are stored. When the grid resolution is reduced, an additional multi-dimensional histogram with B values is stored. The initial phase has therefore $\mathcal{O}(dC + B)$ memory complexity. During the lossy compression phase, only histograms with $\mathcal{O}(S)$ values are materialized as everything else is too big to be part of the final summary. Hence, at most one initial histogram with $\mathcal{O}(B)$ values, $(K + 1)$ digit histograms with $\mathcal{O}(S)$ values and d marginal histograms with C values are stored during the lossy compression. The multiple choice knapsack problem is solved using brute-force, storing only $K = \mathcal{O}(1)$ values. This yields a total memory complexity of $\mathcal{O}(S + B + dC)$ for the construction. □

THEOREM 4 (QUERY TIME). *DigitHist computes an estimate and bounds in $\mathcal{O}(dS)$ time, where d is the data dimensionality and S is the summary size.*

PROOF. The query answer is computed for each of the $K = \mathcal{O}(1)$ digit histograms. First, the z-order bucket indices of two corners of the query region are computed in $\mathcal{O}(d)$. Second, for each dimension four marginal selectivity estimates are computed in $\mathcal{O}(d)$. Then, each multi-dimensional bucket is processed in $\mathcal{O}(d)$ and each bucket is processed at most once resulting in $\mathcal{O}(dS)$. This gives a total query time complexity of $\mathcal{O}(dS)$. □

6. EXPERIMENTAL EVALUATION

6.1 Setup and Datasets

For the experiments we used a machine with an Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz. The data was stored as a binary file on a local hard drive with 10k rpm. All algorithms have been implemented in C++ by the same author and were compiled with GCC 4.9.2 using -O3. Each algorithm runs on a single core and can use up to 2GB of main memory during the construction.

Datasets. The following three datasets are used in the experiments. OSM: a spatial dataset from the OpenStreetMap project that records 2.9 billion two-dimensional GPS-coordinates. HIGGS: a scientific dataset with $11 \cdot 10^6$ 7-dimensional entries from CERN that record particle physics experiments to detect the HIGGS particle in the standard model. ZIPF: a synthetic dataset with 1000 cluster centers and a total of 10^7 points that are distributed according to a Zipf distribution between clusters in all dimensions, i.e., some clusters have much more points than others. The cluster centers are uniformly distributed, and the cluster members are normally distributed around the center. For this dataset we vary the dimensionality from 2 to 16.

Compared Approaches. For the construction of the DigitHist summary, we use $K = 4$ digit histograms and spend 75% of the summary size on the multi-dimensional histogram and 25% on marginal histograms. The initial marginal histograms are equi-width histograms with 2^{25} buckets ($C = 2^{25}$). The initial multi-dimensional histogram has at most 2^{62} buckets ($G = 62$) and at most $B = 2^{25}$ materialized (non-empty) buckets to fit into the available 2GB of main memory.

We compare DigitHist (DH) to the following baseline and state-of-the-art approaches: Equi-width (EW) is a widely known technique that counts the number of points in each cell of a regular grid and uses cumulative counts to achieve querying in 2^d look-ups. Equi-depth (ED) [14] finds buckets such that they count roughly the same number of points. To achieve tighter bounds, our implementation tries to avoid very stretched buckets and keeps track of deviations from the average bucket counts. MinSkew (MSK) [2] is a histogram approach that greedily merges cells to create buckets. In our implementation, we use adaptive refinement as described by the authors and a modified metric as suggested in [21], which yields a higher precision. AVI (AVI) [16] uses one-dimensional histograms in combination with the attribute value independence assumption to produce estimates. It is obviously very limited and only serves as a baseline. Random sampling (RS) [7] summarizes the data with a random subset. It is a simple technique that works well in higher dimensionality. GenHist (GH) [5] iteratively flattens the data density function by moving the biggest density bumps into buckets. We follow the authors' instructions on how to choose the parameters. To avoid prohibitive construction costs, we use a 10MB data sample to create the summary. The GK cross-product summary (GK) [18] constructs a nesting of one-dimensional quantile summaries in d data scans, which can be seen as a set of equi-depth histograms with varying numbers of quantiles per dimension. Since each query may only intersect a sub-linear number of quantiles, the approach provides an (absolute) error guarantee for all queries. The Wei-Yi summary (WY) [20] is a specialized two-dimensional data summary very similar to GK, but it is constructed by sorting the data multiple times. For larger datasets, our implementations of ED and WY violate the 2GB main memory constraint, because they sort the whole dataset in main memory.

Methodology. To measure the precision of summaries, we follow the same methodology as most of the related works by measuring the precision of a biased set of queries. We use workloads of 1000 queries between 1% and 5% selectivity. The queries are centered at randomly chosen data points to simulate real-world queries, prioritizing denser data areas. Note that neither the query centers nor their volumes are uniformly distributed because centers follow the data distribution and, with fixed selectivities, the query volume depends on the data density around the center. As error measures we use the average relative estimation error and the average width of bounds over the set of 1000 queries. The relative estimation error is the absolute difference between the summary's selectivity estimate and the true selectivity, divided by the true selectivity. The relative width of bounds is the absolute difference between the summary's lower and the upper bounds, divided by the true selectivity. Also, we measure the maximum estimation error and maximum width of bounds. Due to space limitations, we only report a short summary of these results at the end of the section.

6.2 Experiments

DigitHist Compression and Marginal Histograms. In the first experiment, we evaluate the effect of the core features of DigitHist, namely digit histogram compression and the use of marginal histograms, on the selectivity error and the width of bounds. For a given summary size, the digit histogram compression allows us to use a higher grid resolution for high density regions, and the marginal histograms allow us to keep track of one-dimensional trends to obtain more accurate estimates of the spread of points inside buckets. We use the ZIPF dataset and vary the number of dimensions to see the impact of data dimensionality. Figure 12 shows that, for a given summary size of 100kB, not using digit histogram compression and using only one digit histogram (= naive compression) would, independently of the data dimensionality, vastly increase the selectivity error up to 3 times, but only slightly decrease the query time by up to 0.1 ms. Disabling the marginal histograms

(= without marginals) increases the selectivity error by 1.5 times for more than two dimensions, but it has only a negligible impact on the query time. This indicates that the digit compression has a favorable impact on precision for all values of dimensionality, whereas the marginal histograms tend to be more important for data with a higher dimensionality, which allows DH to deal with higher dimensional data than previous histogram approaches.

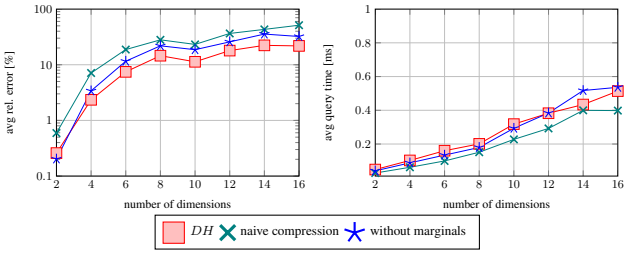


Figure 12: Contribution of DigitHist's core features.

We run the same experiment on the two real-world datasets. For the two-dimensional OSM dataset we got the same result as for the ZIPF data with two dimensions. For the 7-dimensional HIGGS dataset, the effects of using marginal histograms is the same, but the compression has a lower impact for ZIPF with 7 dimensions. The reason is that HIGGS contains very small clusters that can be already precisely captured by one sparsely populated high-resolution grid. For all these experiments, we also measured the impact on the width of bounds, and we got the same behavior as for the selectivity error.

Construction Time. Next, we show the impact of DH's core features on the construction time. The results are summarized in Table 2. The construction time is measured by recording the wall clock time at the beginning and the end of the summary construction. The core features have no significant impact on the construction time for larger datasets. DigitHist's construction time is significantly slower than AVI and EW, but operates in the same time complexity. Random sampling has a very small construction time, while ED, WY and GK have comparably large construction times.

Table 2: Total construction time for 100kB and 1MB summaries.

datasets	OSM (2D - 46.4GB)		HIGGS (7D - 616MB)	
summary size	100kB	1MB	100kB	1MB
DH	16 mins	17 mins	35 secs	3 mins
without marginals	16 mins	17 mins	35 secs	3 mins
naive compression	16 mins	16 mins	7 secs	8 secs
AVI	3 mins	3 mins	2 secs	3 secs
EW	5 mins	5 mins	2 secs	3 secs
ED	2.2 hours	2.5 hours	29 secs	34 secs
WY	2.5 hours	3.1 hours	-	-
GK	1 hour	2.1 hours	20 mins	40 mins

The marginal histograms and compression only add a very small query and construction time overhead, but they bring large improvements in selectivity estimation and width of bounds.

Summary Size. In this experiment, we compare the selectivity error, width of bounds, and query time of different approaches with the same summary size. The three vertical plots in Figure 13a show the result for the OSM dataset and in Figure 13b for the HIGGS dataset. We can see that for the low-dimensional OSM dataset, given the same summary size, DH beats all the other approaches

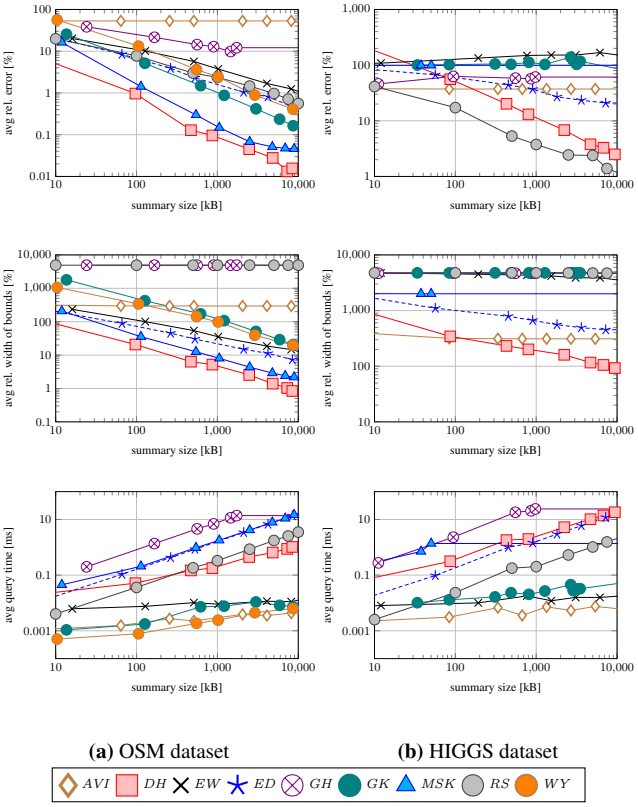


Figure 13: Varying summary size on real-world datasets.

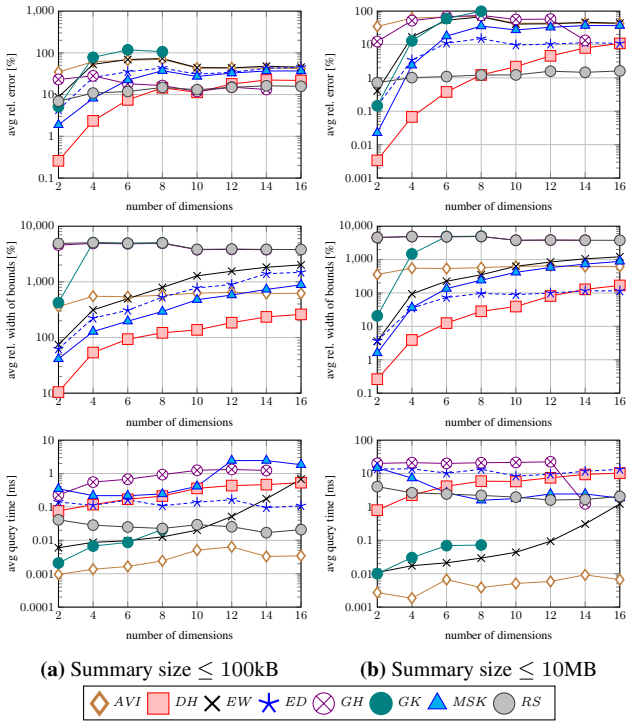
in terms of selectivity error and width of bounds. The approaches EW, AVI, GK and WY are noticeably faster, but are in turn much less precise. AVI does well for low precision, but increasing the summary size does not help beyond a certain point, because it is limited to one-dimensional information. EW needs excessively large summaries to compete with much smaller DH summaries in terms of precision. For test purposes we constructed a 16GB EW summary for the OSM dataset, but it still had a slightly larger estimation error than a 10MB DigitHist summary. We also constructed a 47MB GK summary to see how it behaves with more space. The result was the same, i.e., the average estimation error was slightly higher than for a 10MB DigitHist summary, but it took 5 hours to construct the GK summary, whereas the 10MB DigitHist summary was constructed in half an hour. We observed the same behavior for a 120MB WY summary, constructed in 3.7 hours using more than 46GB of main memory.

For the higher dimensional HIGGS dataset, RS offers the lowest selectivity error, and DH offers the tightest bounds and a lower selectivity error than other histogram approaches. The relative width of bounds of RS and GH exceed 1000%, because their lower bound is close to 0% and their upper bound close to 100% selectivity. Compared to DH, all other histogram approaches, such as AVI, EW, ED and MSK, get much worse in terms of selectivity error and/or width of bounds for this higher-dimensional dataset. The same holds for GK. To better understand how GK performs for larger summaries, we constructed a 88MB GK summary in 32 hours. It has a larger average/maximum estimation error/width of bounds than a 1MB DH summary constructed in 3 minutes.

Regarding query time, approaches that are faster for this dataset have a much lower precision in terms of selectivity error or width of bounds. For the HIGGS dataset, we also note that MSK shows

first an increase in query time for an increasing summary size, but then it remains constant. The reason for this is that this state-of-the-art approach does not increase the summary size anymore from $\approx$ 80kB onwards since it cannot identify more information to store.

Data Dimensionality. We now compare the approaches for different data dimensionality. We use the ZIPF dataset and vary the dimensionality for two different summary sizes. The three vertical plots in Figure 14a show the result for summary sizes of at most 100kB, i.e., all approaches may take up to 100kB of space, and Figure 14b for summary sizes of at most 10MB for all approaches. GH is not included in the experiments with 10MB summaries, because its construction costs become prohibitive when constructing it on a data sample larger than 10MB. GK is included only for up to eight dimensions, because for higher dimensionality the construction time becomes excessive, and the approach already struggles with eight dimensions.



(a) Summary size $\leq$ 100kB **(b)** Summary size $\leq$ 10MB

Figure 14: Varying dimensionality with limited summary size.

We can observe that DH beats the other histogram approaches in selectivity error and width of bounds for all cases, except that ED has in sixteen dimensions slightly tighter bounds with 10MB summaries. The selectivity error of RS remains constant, while the error of the histogram approaches increases with more dimensions until sampling overtakes them. RS overtakes existing histogram approaches at six dimensions and the proposed DigitHist approach at around ten dimensions. Also for this case EW, GK and AVI have the lowest query time, but they provide a very low precision, too. The query time of EW is exponential in the number of dimensions, since it has $\mathcal{O}(2^d)$ time complexity.

In the next experiment, we compare the different approaches for varying dimensionality and show the selectivity error and width of bounds for the ZIPF dataset. Instead of fixing the summary size, this time we depict the values for the summaries with a fixed query time of $\approx$ 1ms. The result is shown in Figure 15. We can see that DH outperforms the other histogram approaches by one order of magnitude and in lower dimensionality even by multiple orders of magnitude. For high dimensional data, RS at some point outperforms DH, but RS cannot provide tight bounds, because it only knows about a small fraction of the data.

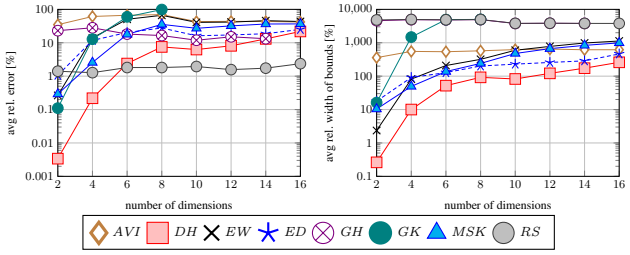
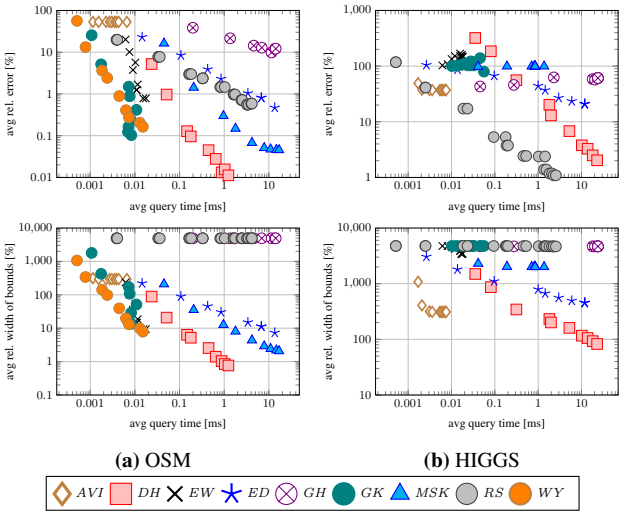


Figure 15: Varying dimensionality with query time $\approx$ 1ms.

Scatterplots Using Query Time. In the next experiment, we analyze the query time. We construct many summaries of different size ($\leq$ 10MB) for each approach and depict a scatterplot for query time with selectivity error and for query time with widths of bounds. The result for the OSM and HIGGS datasets is shown in Figure 16a and Figure 16b, respectively.



(a) OSM **(b)** HIGGS

Figure 16: Comparison of summaries on two real-world datasets.

DH is the only approach to achieve high-precision in all cases. RS achieves lower estimation errors for the high-dimensional dataset, but does not achieve tight bounds like DH. EW is very fast, but reaching the same level of precision would require prohibitively large summaries, i.e., much larger than 10MB. GK is also fast to query and can with 100MB reach a higher precision for the OSM dataset, but requires a construction time of 8 hours.

Worst-case Precision. While in the above evaluation we focused on the average precision, we also measured the maximal relative errors and bounds encountered in the experiments. In general, we observed for all approaches that the maximal error/bounds are about 10 times higher than the average. On the OSM dataset, MSK has smaller maximal estimation errors for summary sizes less than 100KB compared to DH. For more than three dimensions, all approaches except RS and GH reach their limits of applicability and provide maximal estimation errors that could be outperformed by naively estimating 0% selectivity for all queries. This does not apply to the width of bounds, where DH still clearly provides the tightest bounds, but by a thinner margin.

As the maximum errors are very sensitive to outliers, we also looked at the 95-percentiles, i.e., values s.t. 95% of the queries are more precise. The results resemble the averages, indicating that the averages are a good representation of the overall performance.

6.3 Summary

The construction time of DigitHist scales linearly with data size. Summarizing a 46GB dataset takes 16 minutes, while constructing a less precise equi-width summary takes 5 minutes. The construction time also scales linearly with dimensionality and summary size such that larger and higher-dimensional summaries can be constructed in a comparable time frame.

For summaries below 10MB of size and query times below 1 ms, DigitHist delivers the tightest bounds and, up to six dimensions, the highest estimation accuracy. For a lower number of dimensions, DigitHist offers highly accurate estimates and bounds that are tight enough to obviate the need for exact answers in many instances. For example, on the real-world OSM dataset, DigitHist is the only approach to offer bounds with less than 1% relative width. Furthermore, it is also the only approach to offer relative estimation errors below 0.01%. For a higher number of dimensions, DigitHist offers bounds that are tight enough to deduce the order of magnitude of the true selectivity. For instance, on the HIGGS dataset, DigitHist is the only approach to offer bounds with less than 100% relative width.

DigitHist offers many desirable properties and fills the gap of a strong histogram approach that can offer tight bounds and deal with more dimensions, but as expected it does not make all existing techniques obsolete. Equi-width can be preferable if low precision or excessive storage costs are acceptable. Random sampling cannot deduce tight bounds like DigitHist, but the estimation accuracy in high dimensionality is difficult to beat as the approach is not sensitive to data dimensionality. The quantile-based ε-approximation approaches scale poorly with dimensionality, but for two dimensions they can reliably create precise summaries with excellent query times at the cost of a larger summary size and significantly higher construction costs.

7. CONCLUSION

In this paper, we presented a novel summary structure, termed DigitHist, which summarizes data by a set of multi-dimensional histograms, called digit histograms. Each digit histogram represents a different portion of the data stored at a different resolution. Digit histograms are equipped with a one-dimensional marginal histogram for each dimension. They are used to spread the data points inside multi-dimensional buckets. For the construction of DigitHist, we proposed the u-error, which measures a histogram's uncertainty about the data as the expected width of its bounds for a random query region. DigitHist is highly accurate and provides like other histogram approaches bounds for the true selectivity. An in-depth experimental evaluation has shown that DigitHist can cope with more dimensions than other histogram approaches and that for a given summary size, it delivers superior precision than state-of-the-art competitors at a comparable query time.

Future work points in several directions: computing the u-error for non-uniform query distributions, optimizing DigitHist for a given workload distribution, improving the performance in higher dimensionality and adjusting the approach for categorical data.

8. REFERENCES

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at data. In *SIGMOD*, pages 181–192, 1999.

[2] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *SIGMOD*, pages 13–24, 1999.

[3] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: A multidimensional workload-aware histogram. In *SIGMOD*, pages 211–222, 2001.

[4] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[5] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *VLDB J.*, 14(2):137–154, 2005.

[6] P. J. Haas, I. F. Ilyas, G. M. Lohman, and V. Markl. Discovering and exploiting statistical properties for query optimization in relational databases: A survey. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 1(4):223–250, 2009.

[7] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci.*, 52(3):550–569, 1996.

[8] M. Heimel, M. Kiefer, and V. Markl. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In *SIGMOD*, pages 1477–1492, 2015.

[9] Y. E. Ioannidis. The history of histograms (abridged). In *VLDB*, pages 19–30, 2003.

[10] Y. E. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Trans. Database Syst.*, 18(4):709–748, 1993.

[11] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, volume 98, pages 24–27, 1998.

[12] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional selectivity estimation using compressed histogram information. In *SIGMOD*, pages 205–214, 1999.

[13] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD*, pages 448–459, 1998.

[14] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *SIGMOD*, pages 28–36, 1988.

[15] J. M. Phillips. Algorithms for epsilon-approximations of terrains. In *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 447–458. Springer, 2008.

[16] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, pages 486–495, 1997.

[17] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, pages 294–305, 1996.

[18] S. Suri, C. D. Tóth, and Y. Zhou. Range counting over multidimensional data streams. *Discrete & Computational Geometry*, 36(4):633–655, 2006.

[19] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *SIGMOD*, pages 193–204, 1999.

[20] Z. Wei and K. Yi. The space complexity of 2-dimensional approximate range counting. In *SODA*, pages 252–264. SIAM, 2013.

[21] Y. Wu, D. Agrawal, and A. El Abbadi. Using the golden rule of sampling for query estimation. In *SIGMOD*, pages 449–460, 2001.