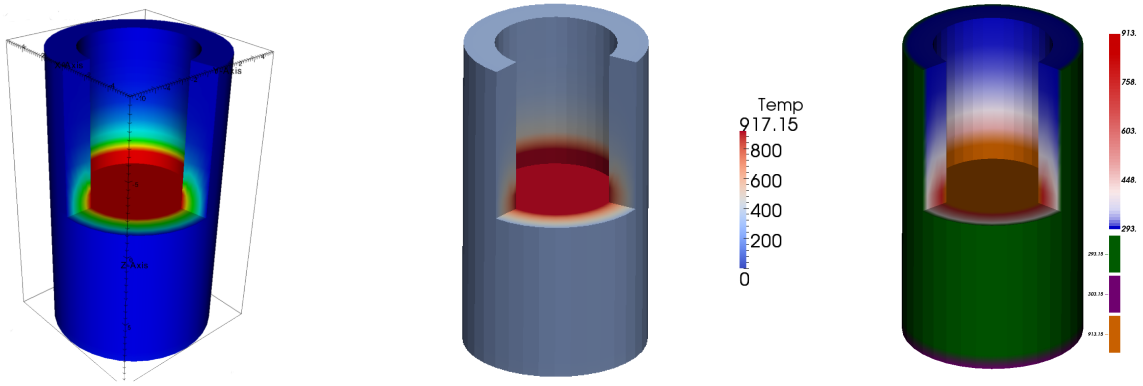
SAND2013-4171C

A Simple and Efficient Sampling Method for Generating Colormaps of Large Data

Category: Technique



(a) VisIt (b) ParaView (c) Prominent values

Fig. 1. Temperature inside a rotating disk reactor. Sampling is able to identify constant boundary conditions and highlight them where they would otherwise be obscured by the full scalar value range.

Abstract— First impressions from initial renderings of data are crucial for directing further exploration and analysis. In most visualization systems, default colormaps are generated by simply linearly interpolating color in some space based on a value's placement between the minimum and maximum taken on by the dataset. We design a simple sampling-based method for generating colormaps that highlights important features. We use random sampling to determine the distribution of values observed in the data. The sample size required is independent of the dataset size and only depends on certain accuracy parameters. This leads to a computationally cheap and robust algorithm for colormap generation. Our approach (1) uses perceptual color distance to produce palettes from color curves, (2) allows the user to either emphasize or de-emphasize prominent values in the data, (3) uses quantiles to map distinct colors to values based on their frequency in the dataset, and (4) supports the highlighting of either inter- or intra-mode variations in the data.

1 Introduction

> Color is the most relative medium in art.
> — Josef Albers, *Interaction of Color*

Color is one of the most prevalent tools used in scientific visualization and possibly the easiest to misuse – knowingly, or, as this paper considers, unknowingly. As Albers notes, and cognitive neuroscience has established experimentally, human perception of color is only relative to its surroundings. In part, at least, this is due to the many orders of magnitude in brightness and contrast our vision must span to be useful in a world with day and night. Using color, it is possible to – within a single image – identify spatial trends at varying scales, discriminate between neighboring values, and even gauge absolute values to some degree.

However, there are very few tools to design the maps between numbers we wish to illustrate and colors that will aid in their undistorted perception[1]. General-purpose visualization tools are often given data with no description of its source, no units of measurement, no accuracy of its computation or measurement, no measurements of its trends, nor any indication how importance is defined. Large datasets that cannot be held in memory (or require a high-performance computer with distributed memory) may not provide easy access to such summary information. The last of these often missing traits – how importance is defined – is of prime significance when choosing a colormap and the least possible to infer without expert knowledge. If commonly-occuring values and patterns are important and variations are noise to be ignored, then colormaps with little change over the scales of unimportant features will be useful so that viewers can focus on important features without distraction. Otherwise, visualizations of the scalar's gradient (for smoothly- varying functions) or alternating colors of high contrast (for both discrete and smooth functions) can bring out important details.

To create more informative colormaps, there are many different data analysis techniques that could be used to determine summary trends in the data. For example, information regarding the relative frequencies of values in the data as well as their spatial relationships in the image could be incorporated into the colormap design. However, analysis techniques that provide detailed information regarding the data could require pre-processing stages, which make them cost-prohibitive when visualizing large data interactively. As a result, default colormaps are typically defined by linearly interpolating dataset function values between the maximum and minimum in the range of the dataset to minimize costs. While computationally inexpensive, this approach disregards the distribution of values observed within the data and incorrectly assumes that the color space is linear. In this paper we introduce an efficient method to generate colormaps via random sampling that addresses both of these issues. Theoretically, our algorithm is simple and provably robust. In practice, with a negligible overhead our algorithm yields images that better highlight features within the data. Most importantly, the required sample size depends only on desired accuracy parameters and is independent of the dataset size, hence scalability is not an issue at all.

Our contributions in detail are as follows:

[1] It is arguably impossible to say that a particular person's perception is biased or unbiased, but it may be possible to quantify how a population's perception of a particular feature is proportional to the evidence for it provided by the data *relative to other features in the same data*.

- We introduce a simple sampling-based algorithm (with a proof of robustness) for identifying important values and ranges in data.
- Given important values in the data, we provide an automated technique for generating discrete and/or continuous color maps.
- We demonstrate results using a number of real and synthetic examples.

The remainder of this paper is organized as follows. In Section 2, we discuss related work and in Section 3 we introduce relevant background and terminology in color theory. We describe our algorithm in Section 4 and provide a proof of robustness in Section 5. Finally, in Sections 6 and 7, we demonstrate results and and discuss conclusions, respectively.

2 Related Work

The most well-known work on color in scientific visualization is Brewer's treatise on the selection of color palettes [3–5]. Her thesis and much surrounding literature focus on choosing palettes that 1) avoid confounding intensity, lightness, or saturation with hue either by co-varying them or using them to convey separate information; and 2) relate perceptual progressions of color with progressions of values in the data to be illustrated or – when the values being illustrated have no relationship to each other – to avoid perceptual progressions of colors so that the image does not imply a trend that is absent in the data. The work of [12] discusses fundamental flaws with the commonly used default "rainbow" colormap and presents results on diverging color maps which have since been adopted by many in the community as they perform much better than the rainbow colormap in scientific visualization settings.

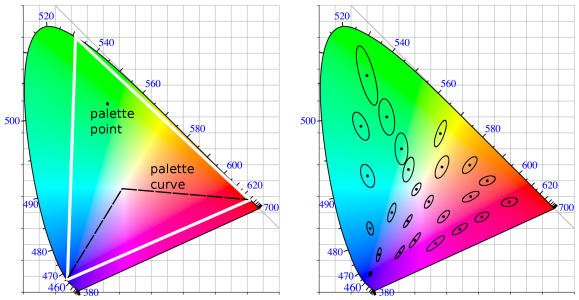
Other significant work has studied the generation of transfer functions for volume rendering. Here, work includes the use of entropy to maximize the "surprisal" and thus the information content of the image [2].The work of [8, 13] compare a number of transfer function generation techniques and provide good high-level overviews of the research in this area.

Finally, tone mapping [9] has been used to adjust images that contain more contrast than their presentation medium is able to provide by modeling how the human visual system deals with contrast.

3 Background

A *color model* is a mathematical abstraction in which colors are represented as tuples of values. Common color models include RGB, CMYK, and HSV. These and other color models differ in how the tuples of values are interpreted and combined to achieve the spectrum of attainable color values. For example, RGB uses additive color mixing, storing values of red, green, and blue. Not all devices can represent and capture colors equally. A *color space* defines which portions of the spectrum of colors can be represented for a particular device. The *gamut* of a device is defined to be the subset of the color space that can be represented, and those colors that cannot be expressed within a color model are considered out of gamut (e.g. red can be expressed in an RGB color space, but cannot be expressed in a CMYK color space). CIELAB is a color space that was created to serve as a device-independent model to be used as a reference. It describes all colors that the human eye can see and is defined by three coordinates: L represents the lightness of a color, a represents its position between green and magenta/red, and b represents its position between yellow and blue.

The distance between colors is often defined in terms of the Euclidean distance between two colors in CIELAB space and is typically referred to as ΔE. Different studies have proposed different ΔE values that have a *just noticeable difference (JND)*. Often a value of $\Delta E \approx 2.3$ is used, however, several variants on the ΔE function have been introduced to address perceptual non-uniformities in the CIELAB color space. These non-uniformities can be visually depicted by *MacAdam ellipses*, which are elliptical regions that contain colors that are considered indistinguishable. These ellipses were identified empirically by MacAdam [10] who found that the size and orientation of ellipses vary widely depending on the test color. Figures 2(a) and 2(b) demonstrate the *chromaticities* (the quality of a color independent of its brightness), visible to the average human. In Figure 2(a) the white triangle is the gamut of the RGB color space and a *palette curve* and *palette point* are shown in this space. In Figure 2(b) multiple MacAdam ellipses are shown to highlight perceptual non-uniformities in the color space.



(a) (b)

Fig. 2. The horse-shoe shape in these figures demonstrates the chromaticities visible to the average human. In (a) the white triangle is the gamut of the RGB color space and a palette curve and palette point are shown in this space. In (b) multiple MacAdam ellipses are shown to highlight perceptual non-uniformities in the color space.

4 Mapping Scalar Values to Color

We introduce a simple algorithm for mapping data values to color according to the distribution of scalar values within the data. We assume the following are provided as input to the mapping algorithm: 1) a list of palette curves and palette points; and 2) a list of prominent values in the data and a cumulative distribution function (CDF) of each range in between prominent values. In Section 5 we describe a provably robust sampling-based approach for the quick estimation of such CDFs and prominent values.

Given the input, we first assign a unique color to each prominent value. Next, we discretize each palette curve p^m into k^m individual palette points of $\Delta E \approx 2.3$ (this value is tunable), see Figure 3(b). As noted in Section 5, each CDF comprises a collection of n disjoint and contiguous intervals, $B_1, \ldots, B_n$. Each B_i has an associated minimum function value f_i^{min} and maximum funtion value f_i^{max}, simply corresponding to the left and right endpoints of B_i. Furthermore, each B_i also has an associated set of samples, whose size is denoted by $s(B_i)$. See the bottom of Figure 3(b), where the relative heights of bars in B_i denote the size of $s(B_i)$.

These intervals/sets represent an approximate CDF in that (roughly speaking), the CDF value at the right endpoint of B_i is given by $\sum_{j=1}^{i} s(B_j) / \sum_{j=1}^{n} s(B_j)$.

Given a scalar value, f, we compute an interpolation factor, f_t, based on the position of f in the CDF. (Again, refer to Figure 3(b).) To do this we identify the bucket B_i that contains f and compute:

$$t_f = \frac{\sum_{j=1}^{i-1} s(B_j) + \left(\frac{f - f_i^{min}}{f_i^{max} - f_i^{min}}\right) * s(B_i)}{\sum_{j=1}^{n} s(B_j)}$$

Once we have identified t_f, we compute the final color, c_f, for a given palette curve, p^m, which has been discretized into k^m palette points as

$$j = t_f * k^m,$$
$$t_j = j - \texttt{floor}(j),$$
$$c_f = c_j + t_j(c_{j+1} - c_j).$$

Implementation details: We allow the user to define both *inter-mode* and *intra-mode* colormaps. In an inter-mode color map, we do not alter the provided palette curves, therefore, for each range between prominent values, we distribute the colors in the provided palette curves according to the distribution of the values within their associated range. When a user wishes to emphasize intra-mode differences within a range between prominent values, we introduce contrast by modifying

the lightness of alternating points along the discretized palette curve in CIELAB space. Furthermore, prominent values can be emphasized or de-emphasized within an image by modifying the lightness of the associated color in CIELAB space. We use the Little CMS color engine [11] to perform all transformations between color spaces and compute ΔE distances between colors.

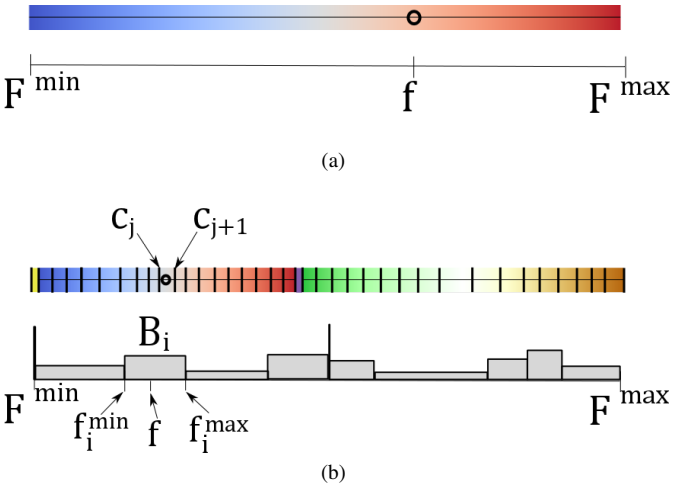


Fig. 3. In figure (a) a simple linear interpolation scheme is depicted. In (b), using our technique, we identify two prominent values and two ranges. Using our CDF-based interpolation scheme we obtain a different color for the function value f.

5 Prominent Values and the Right Intervals

In this section we describe the sampling approaches used to determine prominent values and important subranges of a data set. In general, the default colormaps tend to linearly interpolate between the minimum and maximum values in the data. In other words, they effectively assume that the data distribution is uniform between these values. However, this is not always the case, and we would like to be able to identify significant deviations from this behavior.

We employ two simple sampling algorithms for this purpose. The first algorithm does a direct sampling to determine frequent values in the data set. The second algorithm constructs a series of intervals, such that the frequency of data within each interval is roughly the same. Note that a large variance in the lengths of these intervals indicates a non-uniformity in data value distribution. These intervals represent our approximate CDF. Roughly speaking, for all k, the probability of the data lying in the first k intervals (which is the CDF value upto this point) is proportional to k.

We treat our data as a *discrete* distribution, where for each value r, p_r is the fraction of the data set where the value r is attained (So $\{p_r\}$ describes a distribution over the range of the data set). We use $\mathscr{D}$ to denote this distribution and R to denote the support. For any set S (often an interval of the real line), we use $P(S)$ to denote the probability mass of S.

The analysis of both algorithms follow from straightforward applications of Chernoff bounds. We state the *multiplicative Chernoff bound* (refer to Theorem 1.1 in [7]) for sums of independent random variables.

Theorem 5.1. *[Chernoff bound] Let $X_1, X_2, \ldots, X_k$ be independent random variables in $[0,1]$ and $X = \sum_{i=1}^{k} X_i$.*

- *(Lower tail) For any $\varepsilon > 0$,*

$$\Pr[X < (1-\varepsilon)\mathbf{E}[X]] \leq \exp(-\varepsilon^2 \mathbf{E}[X]/2).$$

- *(Upper tail) For any $\varepsilon > 0$,*

$$\Pr[X > (1+\varepsilon)\mathbf{E}[X]] \leq \exp(-\varepsilon^2 \mathbf{E}[X]/3).$$

- *(Upper tail) For any $t > 2e\mathbf{E}[X]$,*

$$\Pr[X > t] \leq 2^{-t}.$$

In our theorems, we do not attempt to optimize constants. The running time of our algorithms does not depend on the data size, and the theorems are basically proof of concepts. Our empirical work will show that the actual samples sizes required are quite small. For convenience, we use c and c' to denote sufficiently large constants.

5.1 Finding important elements

Our aim is to determine values of r such that $p_r > \tau$ is large, where $\tau \in (0,1)$ is a *threshold parameter*.

`find-important`(s, τ)
Inputs: sample size s, threshold τ
Output: the "frequent set" of range elements, I.

1. Generate set S of s independent random samples from $\mathscr{D}$
2. Initialize important set $I = \emptyset$.
3. For any element $r \in \mathscr{D}$ that occurs more than $s\tau/2$ times in S, Add r to I.
4. Output I.

The following theorem states that (up to some approximation), I is indeed the set of frequent elements. The constants in the following are mainly chosen for presentation. (Instead of $p_r < \tau/8$ in the following, we can set it to τ/α, for any $\alpha > 1$, and choose s accordingly.)

Theorem 5.2. *Set $s = (c/\tau)\ln(c/(\tau\delta))$. With probability $> 1-\delta$ (over the samples), the output of* `find-important`(τ, δ) *satisfies the following.*

If $p_r > \tau$, then $r \in I$.
If $p_r < \tau/8$, then $r \notin I$.

Proof. We first prove that with probability at least $1-\delta/2$, for all $p_r > \tau$, $r \in I$. Then we show that with probability at least $1-\delta/2$, for all $p_r < \tau/8$, $r \notin I$. A union bound then completes the proof.

Consider some r such that $p_r > \tau$. Let X_i be the indicator random variable for the event that the ith sample is r. So $\mathbf{E}[X_i] = p_r$ and all X_is are independent. We set $X = \sum_{i \leq s} X_i$ and apply the Chernoff lower tail of Theorem 5.1 with $\varepsilon = 1/2$. Hence, $\Pr[X < \mathbf{E}[X]/2] \leq \exp(-\mathbf{E}[X]/8)$. Note that $\mathbf{E}[X] > s\tau$. We obtain $\Pr[X < s\tau/2] \leq \Pr[X < \mathbf{E}[X]/2] \leq \exp(-\mathbf{E}[X]/8) \leq \exp(-s\tau/8) \leq \delta\tau/16$. There are at most $1/\tau$ values of r such that $p_r > \tau$. By the union bound, the probability that there exists some such value of r occurring less than $s\tau/2$ times is at most $\delta/16$. Hence, with probability $> 1-\delta/16$, $\forall p_r > \tau$, $r \in I$.

Define set $A = \{r | p_r \geq \tau/8\}$, and $R' = R \setminus A$. The second part is stated as Lemma 5.3 below with $\alpha = \tau/8$. We get that with probability $> 1-\delta/2$, all $r \in R'$ individually occur less than $s\tau/2$ times. Hence, none of them are in I. □

Lemma 5.3. *Let $\alpha \in (0,1)$ and $s > (c/8\alpha)\ln(c/(8\alpha\delta))$. Consider a set R' such that $\forall r \in R'$, $0 < p_r \leq \alpha$. With probability $> 1-\delta/2$, the following holds. For all $r \in R'$, the number of occurrences of r in s uniform random samples from $\mathscr{D}$ is at most $4s\alpha$.*

Proof. We apply Claim 5.4 (given below). This gives a series of intervals $R_1, R_2, \ldots, R_n$, such that for all $m < n$, $P(R_m \cap R') \in [\alpha, 2\alpha]$. Also, $P(R_n \cap R') \leq 2\alpha$.

Consider some R_m for $m < n$, and let Y_i be the indicator random variable for the ith sample falling in R_m. We have $\mathbf{E}[Y_i] \in [\alpha, 2\alpha]$ and $\mathbf{E}[Y] \leq [\alpha s, 2\alpha s]$ (where $Y = \sum_{i \leq s} Y_i$). Since the Y_is are independent, we can apply the first Chernoff upper tail with $\varepsilon = 1$. Hence, $\Pr[Y > 4s\alpha] \leq \exp(-\alpha s/3) = \delta\alpha/3$.

Now focus on R_n and define Y analogous to above. If $P(R_n \cap R') > \alpha/2$, we can apply the previous argument with $\varepsilon = 1$. Again, we deduce that $\Pr[Y > 4s\alpha] \leq \exp(-s\alpha/6) = \delta\alpha/3$. If $P(R_n \cap R') < \alpha/2$, we apply the second Chernoff tail with $t = 4s\alpha$ (observing that $4s\alpha \geq (2e)\alpha/2$) to deduce that $\Pr[Y > 4s\alpha] \leq 2^{-4s\alpha} \leq \delta\alpha/3$.

We apply the union bound over all R_m for $m \leq n$ (at most $1/\alpha + 1$ is number), so the probability that there exists some R_d that appears

more than $4s\alpha$ times is at most $\delta/2$. Hence, with probability at least $1-\delta/2$, no element in R' can appear more than $4s\alpha$ times. □

Claim 5.4. *Let $\alpha \in (0,1)$. Consider a set R' such that $\forall r \in R'$, $0 < p_r \leq \alpha$. There exists a sequence of numbers $\min_{r\in R'} r = z_1, z_2, \ldots, z_k = \max_{r\in R'} r$ ($k \geq 2$) such that for all $i < k-1$, $P([z_i, z_{i+1}) \cap R') \in [\alpha, 2\alpha]$ and $P([z_{k-1}, z_k]) \leq 2\alpha$.*

Proof. This is done through a simple iterative procedure. We start with $z_1 = \min_{r\in R'} r$. For z_i, we describe how to find z_{i+1}. Imagine z_{i+1} initialized to z_i and continuously increased until the $P([z_i, z_{i+1}] \cap R')$ (note that we use a closed interval) exceeds 2α. (If z_{i+1} crosses $\max_{r\in R'} r$, then we have found the last interval and terminate this process.) Now, $P([z_i, z_{i+1}) \cap R')$ (the open interval) must be less than 2α, or we would have stopped earlier. Furthermore, $P([z_i, z_{i+1}) \cap R') = P([z_i, z_{i+1}] \cap R') - p_{z_{i+1}} \geq 2\alpha - \alpha = \alpha$. □

5.2 Finding the right intervals

Our aim is to construct a series of disjoint intervals that (almost) equally partition the probability mass. To gain some intuition, consider a positive integer v and a sequence of numbers $y_0, y_2, \ldots, y_v$ where $y_0 = \min_{r\in R} r$, $y_v = \max_{r\in R} r$, and for all $i < v$, $P([y_i, y_{i+1})) = 1/v$. Our algorithm will try to find these intervals (for a parameter v). Of course, such intervals may not even exist, due to the discrete nature of $\mathscr{D}$. Nonetheless, we will try to find suitable approximations. For this reason, we assume that there are no prominent values in $\mathscr{D}$. (This is an acceptable assumption, since prominent values can be predetermined by `find-important`.)

There are two parameters for `find-intervals`: the sample size s and the block size b. For convenience, assume b divides s. The output is a series of s/b intervals, each with (provably) approximately the same probability mass. As we mentioned earlier, this constitutes an approximation to the CDF, since the probability mass of the first k of these intervals will be (approximately) proportional to k.

`find-intervals`(s,b)
Inputs: sample size s, block size b
Outputs: Intervals $B_1, B_2, \ldots$

1. Generate set S of s independent random samples from $\mathscr{D}$.
2. Sort these to get the (ordered) list $\{x_1, x_2, x_3, \ldots, x_s\}$.
3. Output the intervals $B_1 = [x_1, x_b)$, $B_2 = [x_{b+1}, x_{2b})$, etc. In general, the ith interval B_i is $[x_{(i-1)b+1}, x_{ib})$ and there are s/b blocks. The samples in this interval form the associated set, so $s(B_i) = |B_i \cap S|$.

This main theorem involves some play of parameters, and we express s and b in terms of an auxiliary (integer) parameter v. Again, the constants chosen here are mainly given for some concreteness and notational convenience. We use the notation $A \in (1\pm\beta)B$ as a shorthand for $A \in [(1-\beta)B, (1+\beta)B]$.

Theorem 5.5. *Set $s = cv\ln(cv/\delta)$ and $b = s/v$. Suppose there exists no $r \in R$ such that $p_r > 1/100v$. With probability $> 1-\delta$, the following holds. For each output interval B, $P(B) \in (1\pm 1/20)/v$. Furthermore, $P((-\infty, x_1))$ and $P((x_s, +\infty))$ are at most $1/50v$.*

Proof. We first apply Claim 5.4 with $\alpha = 1/100v$ and $R' = R$, to get the sequence $z_1, z_2, \ldots, z_k$. We have $P([z_i, z_{i+1})) \in [1/100v, 1/50v]$ for $i < k-1$ and $P([z_{k-1}, z_k]) \leq 1/50v$. We prove the following lemma.

Lemma 5.6. *Set $s = cv\ln(cv/\delta)$. Let $Z_{i,j}$ be the number of samples falling in interval $[z_i, z_j)$. With probability at least $1-\delta/2$, for all pairs i, j, $Z_{i,j} \in (1\pm 1/20)sP([z_i, z_j))$.*

Proof. Fix interval $[z_i, z_j)$. Let X_m be the probability of the mth sample falling in $[z_i, z_j)$. We have $\mathbf{E}[X_m] = P([z_i, z_j)) \geq 1/100v$ and all X_m's are independent. Also, $Z_{i,j} = \sum_{m\leq s} X_m$, and $\mathbf{E}[Z_{i,j}] \geq s/100v$. The upper Chernoff tail yields $\Pr[Z_{i,j} < (1-1/20)sP([z_i, z_j))] \leq \exp(-sP([z_i, z_j))/800)$. By a union bound over both tails and plugging in the bound $P([z_i, z_j)) \geq 1/100v$, the probability that $X_{i,j} \notin (1\pm 1/20)sP([y_i, y_j))$ is $\exp(-s/(800\cdot 100v)) + \exp(-sP([y_i, y_j))/(1200\cdot 100v))$. Doing the calculations (for sufficiently large constant c), this probability is at most δ/cv^2. A union bound over all intervals (at most $\binom{100v+1}{2}$ of them) completes the proof. □

We apply Lemma 5.6, so with probability $1-\delta/2$, for all i, j, $Z_{i,j} \in (1\pm 1/10)sP([y_i, y_j])$. Now, we apply Lemma 5.3 with $\alpha = 1/100v$. With probability $> 1-\delta/2$, no element occurs more than $s/25v$ times. By a union bound, both these hold with probability $> 1-\delta$. Under these conditions, we complete our proof.

Fix a block B, and let $[y_i, y_j)$ be the smallest such interval that contains B. Let $[y_{i'}, y_{j'})$ be the largest such interval inside B. Observe that $P([y_i, y_j)) - 2\cdot(1/50v) \leq P(B) \leq P([y_i, y_j))$ and $P([y_{i'}, y_{j'})) \leq P(B) \leq P([y_{i'}, y_{j'})) + 2\cdot(1/50v)$.

Let ℓ denote the number of sample points in $B = [x_h, x_{h+1})$. Obviously, $\ell \leq b$, by construction. Also, ℓ is at least b minus the number of sample occurrences of x_{h+1}. So $\ell \geq b - s/25v = (1-1/25)s/v$. Since $\ell \leq X_{i,j}$, we get $(1-1/25)s/v \leq (1+1/20)sP([y_i, y_j))$. Since $\ell \geq X_{i',j'}$, $s/v \geq (1-1/20)sP([y_{i'}, y_{j'})$. Relating the probabilities to $P(B)$, we get $(1-1/25)/v \leq (1+1/20)(P(B)+1/25v)$ and $1/v \geq (1-1/20)(P(B)-1/25v)$. Rearranging the terms, we complete the proof for output interval B.

No samples fall in the intervals $(-\infty, x_1)$ and $(x_s, +\infty)$. Hence, they cannot completely contain any interval of the form $[y_i, y_{i+1}]$, and their probabilities are at most $1/50v$.

□

6 Results

In this section we demonstrate the results of applying our colormap generation algorithm to a variety of datasets. These include the Mandelbrot dataset, a rotating disk reactor, and two combustion data sets. We compare images of the data created with default colormaps of several visualization tools to colormaps created using our technique.

The Mandelbrot dataset is a synthetic function that can be sampled at any resolution. By contouring the volumetric function using a geometric sequence of isovalues, we obtained auxiliary data used to characterize how prominent value detection behaves as distinct values become near enough to appear as a continuous range, see Figure 6. We also run the algorithm on the data without contouring in Figure 7.

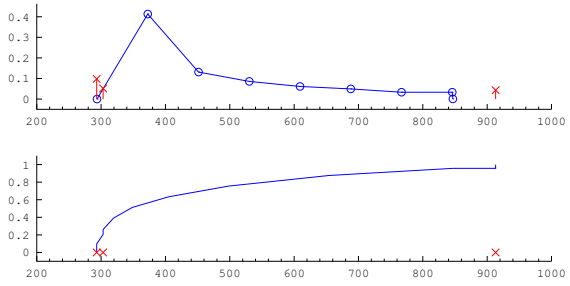


Fig. 4. The probability density function (PDF, top) and cumulative density function (CDF, bottom) of the rotating disk reactor, shown in blue and obtained from the prominent values and blocked ranges. Prominent values and their probabilities are shown in red.

The rotating disk reactor, shown in Figure 1, is a steady-state simulation of continuous vapor deposition (CVD) carried out by MP-Salsa [14, § D.2]. A notch has been removed for illustrative purposes. It is a small dataset that contains thermodynamic state variables, velocity, and chemical species concentrations at each node. The simulated region is a fluid-filled cavity between a concentric tube and rod where reactants flow up from the unobstructed volume (bottom), across the heated rod, and exit at the annulus (top). Of interest is the fact that temperature boundary conditions have been imposed: the outer cylindrical wall is held at an ambient temperature of 293.15 K, the inner rod cap is heated to 913.15 K, and the fluid entering the chamber is a constant 303.15 K. Because the two lower temperature conditions are very near each other relative to the heated rod cap, it is impossible to distinguish them in the default views of ParaView and VisIt. However,

by passing the temperature through the sampling algorithm above, we obtain the PDF and CDF shown in Figure 4 , along with associated prominent values that pull out these features. By assigning colors outside the prominent range to these prominent values, the difference is apparent.

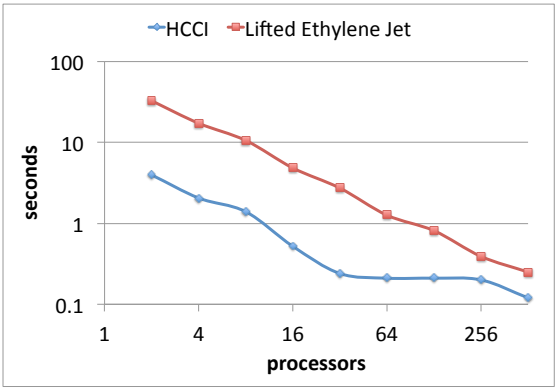
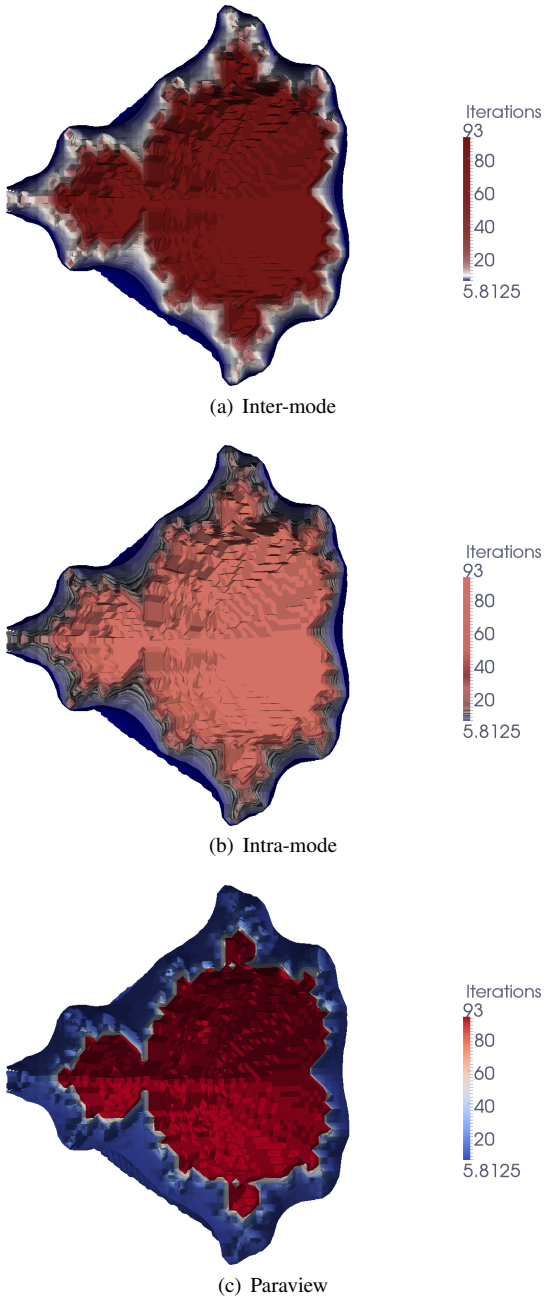


Fig. 5. Our technique demonstrates good scalability as the number of samples required by our algorithm is fixed according to accuracy parameters as opposed to data set size.

In addition to these two datasets we also demonstrate our results on two combustion data sets: HCCI, and Lifted Ethylene Jet. These datasets were generated by S3D [6], a turbulent combustion simulation code that performs first principles-based direct numerical simulations in which both turbulence and chemical kinetics introduce spatial and temporal scales spanning typically at least 5 decades. The HCCI dataset is a study of turbulent auto-ignitive mixture of Di-Methyl-Ether and air under typical Homogeneous Charge Compression Ignition (HCCI) conditions [1]. This simulation is aimed at understanding the ignition characteristics of typical bio-fuels for automotive applications and has a domain size of over 175 million grid points. The second simulation describes a lifted ethylene jet flame [15], involved in a reduced chemical mechanism for ethylene-air combustion, with a domain size of 1.3 billion grid points.

Figure 7 contains images comparing our technique with that of Paraview for the Mandelbrot, HCCI, and Lifted Ethylene Jet data sets. The first row demonstrates default color maps generated by Paraview for the Mandelbrot, HCCI, and Lifted Ethylene Jet data sets. The second and third row contrast inter- vs. intra-mode differences with the prominent values emphasized as colors with low lightness in CIELab space, while the fourth and fifth row contrast inter- vs. intra-mode idfferences with prominent values de-emphasized as colors with high lightness in CIELab space. In particular, within the combustion datasets, our approach visually depicts structures in the data that are difficult to see in the default view. These structures cover a small relatively small percentage of overall scalar domain, however these values are observed with relatively high frequency within the data. We note that our approach can be effective at identifying coherent structures in spite of the fact that we are not integrating spatial information into our estimates.

Scalability: Figure 5 highlights the scalability of our approach for the Lifted Ethylene Jet and HCCI data sets. We performed our experiments on Lens at the Oak Ridge Leaership Computing Facility. Lens is a 77-node commodity-type Linux cluster whose primary purpose is to provide a conduit for large-scale scientific discovery via data analysis and visualization of simulation data. Lens has 45 high-memory nodes that are configured with 4 2.3 GHz AMD Opteron processors and 128 GB of memory. The remaining 32 GPU nodes are configured wiht 4 2.3 GHz AMD Opteron processors and 64 GB of memory.



(a) Inter-mode

(b) Intra-mode

(c) Paraview

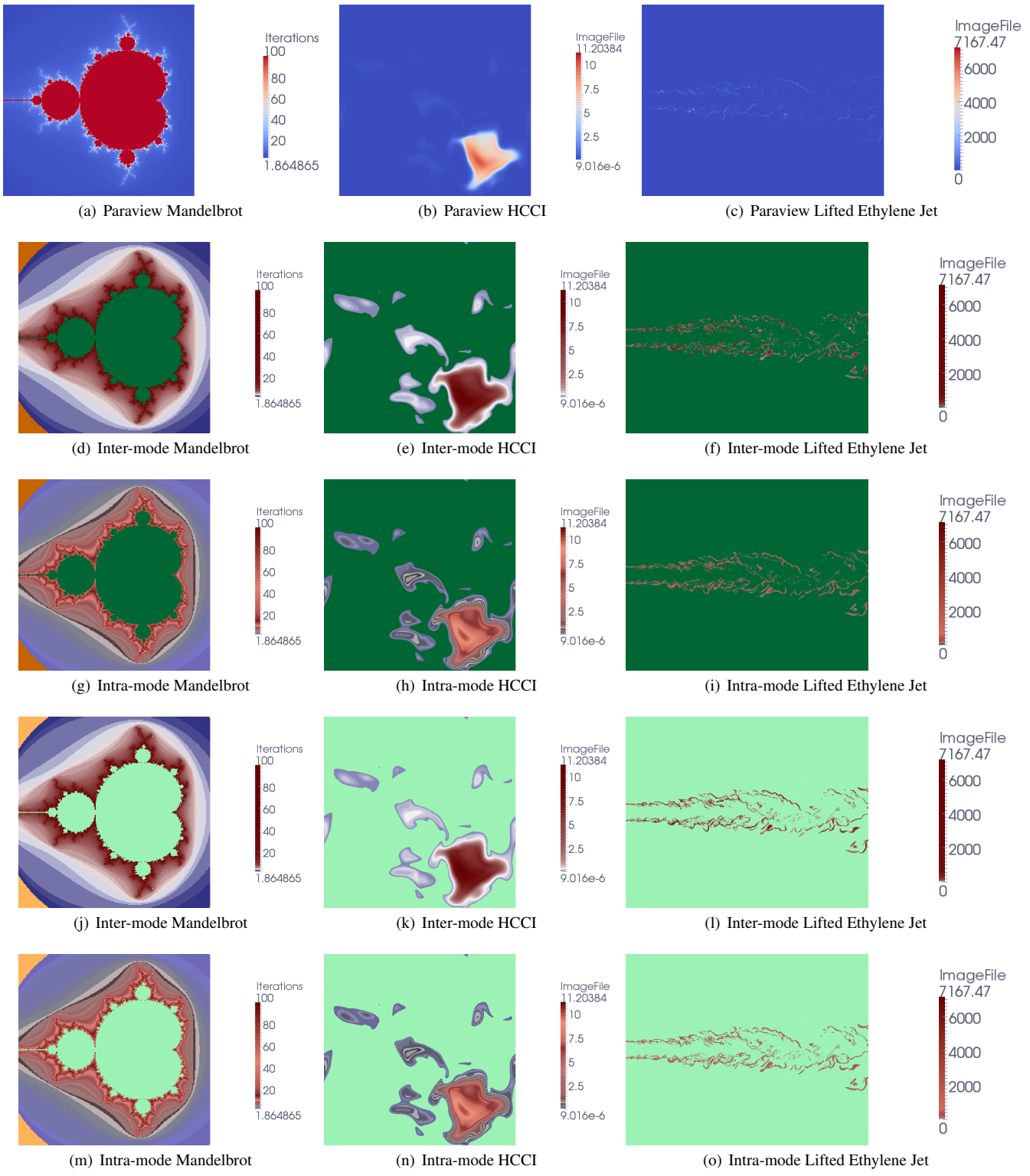
Fig. 6. These images show 32 contour values, each of which is identified by our algorithm as a prominent value. Many of these isovalues lie closely together and are difficult to differentiate using traditional default color maps. Using our algorithm it is much easier to see that there are in fact many individual surfaces.

7 Conclusion

We introduced a sampling based method to generate colormaps. Our algorithm identifies important values and ranges in data, and uses these values to automatically generate discrete and/or continuous color maps. Our experiments showed the new colormaps yields images that better highlight features within the data. The proposed approach is simple and efficient, yet provably robust. Most importantly, the number of samples required by the algorithm depends only on desired accuracy in estimations and is independent of the dataset size. This provides excellent scalability for our algorithms, as the required preprocessing time remains constant despite increasing data as we move to exascale computing. The algorithms are also efficiently parallelizable and and yield linear scaling.

References

[1] G. Bansal. *Computational Studies of Autoignition and Combustion in Low Temperature Combustion Engine Environments*. PhD thesis, Univer-



(a) Paraview Mandelbrot (b) Paraview HCCI (c) Paraview Lifted Ethylene Jet

(d) Inter-mode Mandelbrot (e) Inter-mode HCCI (f) Inter-mode Lifted Ethylene Jet

(g) Intra-mode Mandelbrot (h) Intra-mode HCCI (i) Intra-mode Lifted Ethylene Jet

(j) Inter-mode Mandelbrot (k) Inter-mode HCCI (l) Inter-mode Lifted Ethylene Jet

(m) Intra-mode Mandelbrot (n) Intra-mode HCCI (o) Intra-mode Lifted Ethylene Jet

Fig. 7. The first row demonstrates default color maps generated by Paraview for the Mandelbrot, HCCI, and Lifted Ethylene Jet data sets. The second and third row contrast inter- vs. intra-mode differences with the prominent values emphasized as colors with low lightness in CIELab space, while the fourth and fifth row contrast inter- vs. intra-mode idfferences with prominent vlaues de-emphasized as colors with high lightness in CIELab space.

sity of Michigan, 2009.

[2] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Vis '05: Proceedings of the IEEE Visualization 2005*, pages 487–494, 2005.

[3] C. A. Brewer. Guidelines for selecting colors for diverging schemes on maps. *The Cartographic Journal*, 33(2):79–86, 1996.

[4] C. A. Brewer. A transition in improving maps: The ColorBrewer example. *Cartography and Geographic Information Science*, 30(2):159–162, 2003.

[5] C. A. Brewer, G. W. Hatchard, and M. A. Harrower. ColorBrewer in print: A catalog of color schemes for maps. *Cartography and Geographic*

Information Science, 30(1):5–32, 2003.

[6] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorski, R. Sankaran, S. Shende, and C. S. Yoo. Terascale direct numerical simulations of turbulent combustion using s3d. *Computational Science and Discovery*, 2:1–31, 2009.

[7] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[8] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[9] G. Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4), Dec. 1997.

[10] D. L. MacAdam. "visual sensitivities to color differences in daylight". *JOSA*, 32(5):247–274, 1942.

[11] M. Maria. Little CMS: How to use the engine in your applications, 2012.

[12] K. Moreland. Diverging color maps for scientific visualization. In G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. S. Coming, editors, *Advances in Visual Computing, 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30 - December 2, 2009, Proceedings, Part II*, volume 5876 of *Lecture Notes in Computer Science*, pages 92–103. Springer, 2009.

[13] H. Pfister, W. E. Lorensen, W. J. Schroeder, C. L. Bajaj, and G. L. Kindlmann. The transfer function bake-off (panel session). In *IEEE Visualization*, pages 523–526, 2000.

[14] A. Salinger, K. Devine, G. Hennigan, H. Moffat, S. Hutchinson, and J. Shadid. MPSalsa: A finite element computer program for reacting flow problems, part 2 – users guide. Technical Report SAND96-2331, Sandia National Laboratories, Sept. 1996.

[15] C. S. Yoo, R. Sankaran, and J. H. Chen. Three-dimensional direct numerical simulation of a turbulent lifted hydrogen jet flame in heated coflow: Flame stabilization and structure. *Journal of Fluid Mechanics*, 640:453–481, 2009.