First Considerations in Computing and Using Hypersurface Curvature for Energy Efficiency

Jacob D. Hauenstein
The University of Alabama in Huntsville
301 Sparkman Drive
Huntsville, AL 35899
jacob.hauenstein@uah.edu

Timothy S. Newman
The University of Alabama in Huntsville
301 Sparkman Drive
Huntsville, AL 35899
timothy.newman@uah.edu

ABSTRACT

Energy consumption for computing and using hypersurface curvature in volume dataset analysis and visualization is studied here. Usage in both the base case and when more energy-optimal strategies, particularly computational (especially for linear algebraic steps) strategies, are the primary foci here and are considered for analysis tasks that are precursors to visualization. Compilation-based effects on energy usage are a secondary focus. Efforts here are on Intel x86, which is popular and has power measurement capabilities. Additionally, a first-time visualization of hypersurface curvature distributions in a brain imaging scenario is exhibited. The work aims to advance understanding of computing's energy footprint and to provide guidance for energy-responsible volume data analysis.

Keywords

Power Efficiency, Green Computing, Curvature, Volumetric Data, Hypersurface

1 INTRODUCTION

Energy consumption of data processing activities has recently received increased attention from policy-makers, data center deployment planners, and computing researchers. Methods to characterize computing energy use thus has been one direction for energy-related research in computing, with characterization still an early-stage technology [Lan23]. In this paper, one of our focuses is characterizing energy use for certain computations related to analysis and visualization. Some computing environments, like recent Intel x86 CPUs (our focus here), have built-in power-monitoring features that are accessible for inspection by software developers, making such characterizations accessible to interested parties. Other environments do not well-expose their power usage, making characterization more challenging for them. (Some reports involving use of external monitoring systems to measure power consumption do exist (e.g., [Lin19]), though, and generic estimators based on memory and core use also exist (e.g., [Lan21]).) Some of the characterizations have involved system-level considerations, especially for large data centers, with those characterizations sometimes employed in setting purchase specifications. A characterization of server energy use has been reported by Fuchs et al. [Fuc20], for example. Additionally, characterizing energy from manufacturing of computing systems has also been considered [Gup22]. One other area of investigation has considered characterizations of competing computing paradigms, such as opportunities using FPGA computation (e.g., [EM20]). Comparative analyses of many instances of a class of algorithms have also been reported (e.g., [Hen20]).

Finally, another research theme has been creation and examination of strategies that a given type of algorithm can use to reach its solution in a more energy-optimal manner. Such approaches offer great promise–they are perhaps the "most productive...green computing activity" [Lan21]. They can be pursued either in an end-device agnostic manner or with focus on a specific computing device. Beckitt-Marshall [BM21], for example, has examined compiler optimization strategies to determine best optimization settings to achieve low energy usage for two file compression approaches, an ambient occlusion renderer, and several other algorithms on a widely-used type of CPU. Another strategy has been to exploit cache properties to reduce energy usage (e.g., [Tit15]).

The effort here both characterizes power usage and explores energy-efficient strategies for a class of volume data analysis algorithms, hypersurface curvature determination, and one volume visualization scheme, maximum intensity projection, run on Intel x86 (one of the most popular end computing environments for volume

data analysis). (Hypersurface curvatures are discussed in more detail in Section 2.) As one of the first such efforts for the volume data analysis arena, a limited series of strategies is considered here, although they are applicable to many other volume data analysis and visualization approaches.

2 BACKGROUND

This section presents some necessary background. This includes details about hypersurface curvature (including its utility and the mathematics for computing it) and details about the four hypersurface curvature determination methods considered in our energy usage experiments (presented later). Finally, some details about the aims and motivations of the work are presented.

2.1 Hypersurface Curvature

Curvature measures of 3D manifolds (often called *hypersurfaces* [Mon92]) have been found to be very useful descriptors for a variety of tasks, especially in the fields of geology (where such curvature measures are useful for identifying faults or fractures [Bra10] and for visualizing seismic phenomena [Ald14]) and medicine (where such curvature measures are useful for classification of tumors [Hir18] and arterial measurement [Suz18]). For such tasks, the three principal curvature values, denoted κ_1, κ_2, and κ_3 (ordered such that $\kappa_1 > \kappa_2 > \kappa_3$), are of value.

Hypersurface curvatures are also useful for surface reconstruction [Pap07] and for classification of data points within a volume, where such points can be classified based on the relative, relative absolute, and average values of these three principal curvature values [Hir01].

Formally, hypersurface curvature can be defined as follows. Let (u,v,w) denote a grid (or sample) point within a scalar volume (where $0{\leq}u{<}N_u$, $0{\leq}v{<}N_v$, $0{\leq}w{<}N_w$ for a volume of size $N_u{\times}N_v{\times}N_w$). The value at point (u,v,w) is then denoted $f(u,v,w)$ with f representing the underlying function that generates the volume and f_u representing the partial derivative of f in the u direction. The hypersurface's three principal curvatures (i.e., of f) are then the eigenvalues of the matrix:

$$\frac{1}{l}\begin{bmatrix} f_{uu} & f_{uv} & f_{uw} \\ f_{uv} & f_{vv} & f_{vw} \\ f_{uw} & f_{vw} & f_{ww} \end{bmatrix} \begin{bmatrix} 1+f_u^2 & f_u f_v & f_u f_w \\ f_u f_v & 1+f_v^2 & f_v f_w \\ f_u f_w & f_v f_w & 1+f_w^2 \end{bmatrix}^{-1}, \tag{1}$$

where

$$l = \sqrt{1+f_u^2+f_v^2+f_w^2}. \tag{2}$$

Thus, computation of κ_1, κ_2, and κ_3 requires knowledge of the first derivatives, second derivatives, and mixed partial derivatives of f. Often, the continuous form of f is unknown, because the data under consideration was acquired via a sensor. In such cases, these necessary derivatives must be estimated in order to calculate κ_1, κ_2, and κ_3 from Eqn. 1.

2.2 Hypersurface Curvature Determination Methods

Our studies consider four methods for determining hypersurface curvature from volumetric data. These methods all work by first estimating the necessary derivatives and then computing the hypersurface curvature using Eqn. 1. These four methods have previously been comparatively studied on the bases of their accuracy and run time [Hau21]. Because the methods all differ in their derivative estimation approach, all four methods exhibit varying run times, accuracies, and energy usages. Here, we briefly describe the four methods considered.

2.2.1 Taylor Exp.-based Conv. Kernels (TE)

One hypersurface curvature determination method, denoted **TE**, estimates derivatives using convolution. Specifically, it uses convolution filters derived from the Taylor Expansion along each axis to estimate all the necessary derivatives. Once these derivatives are known, the three principal curvatures are computed as the eigenvalues of Eq. 1. The **TE** approach has previously been used in determination of both surface curvature [Kin03] and hypersurface curvature [Hau19], where it was found to be a among the fastest approaches.

The method's filters are determined according to a framework that allows construction of filters with arbitrary accuracy and continuity. For our experiments with the **TE** approach, we used filters with C^3 continuity and fourth order accuracy, following prior works (e..g., [Kin03]). At these continuity and accuracy parameters, the first derivative kernel is size 5 and the second derivative kernel is size 7.

2.2.2 B-Spline-based Derivatives (BS)

The **BS** hypersurface curvature determination method uses f as the coefficients of a tricubic B-Spline [Hau19]. This B-Spline represents a continuous form that approximates f, and the derivatives of that continous form, in conjunction with Eq. 1, are used to determine the principal curvatures.

It is possible to configure the B-Spline in a number of ways (e.g., with varying numbers of knots or varying degree). For our experiments , we configured the B-Spline similarly to other reports–with knot count the same as input volume dimensions.

2.2.3 Orthogonal Polynomials-based Convolution Kernels (OP)

Another hypersurface curvature method, denoted **OP**, also uses convolution to estimate the necessary derivatives and then computes the resulting curvatures via Eq. 1. **OP** uses convolution kernels derived from orthogonal polynomials.

The **OP** method uses kernels of odd size N that are generated according to three functions $b_0(x)$, $b_1(x)$, and $b_2(x)$:

$$b_0(x) = \frac{1}{N}, \tag{3}$$

$$b_1(x) = \frac{3}{M(M+1)(2M+1)}x, \tag{4}$$

$$b_2(x) = \frac{1}{P(M)}(x^2 - \frac{M(M+1)}{3}), \tag{5}$$

with $M = \frac{N-1}{2}$. $P(M)$ is:

$$P(M) = \frac{8}{45}M^5 + \frac{4}{9}M^4 + \frac{2}{9}M^3 - \frac{1}{9}M^2 - \frac{1}{15}M. \tag{6}$$

From these kernels, derivatives are estimated via:

$$a_{ijk} = \sum_{u,v,w \in m \times m \times m} f(u,v,w) b_i(u) b_j(u) b_k(u), \tag{7}$$

where $m = \{\frac{-(N-1)}{2}, ..., \frac{(N-1)}{2}\}$. For example, the estimate f_u would be found using a_{100}.

Aside from one set of supplemental experiments at the end of Section 4, all of our experiments follow prior works (e.g., [Hau19]) and use $N = 7$.

Since convolution with such kernels implicitly does a least squares fitting [Fly89], OP results match those of a linear regression-based local surface fitting (without explicitly performing any linear regression).

2.2.4 Deriche Filter-based Convolution Kernels (DF)

The **DF** hypersurface curvature determination method also uses convolution to estimate derivatives and then computes curvature via Eq. 1. It uses convolution kernels based on three Deriche Filters ($\hat{f}_0$ for smoothing and $\hat{f}_1$, $\hat{f}_2$ to estimate the first and second derivatives, respectively) [Der90]. $\hat{f}_0$, $\hat{f}_1$, and $\hat{f}_2$ are defined as:

$$\hat{f}_0(x) = c_0(1 + \alpha|x|)e^{-\alpha|x|}, \tag{8}$$

$$\hat{f}_1(x) = -c_1 x \alpha^2 e^{-\alpha|x|}, \tag{9}$$

$$\hat{f}_2(x) = c_2(1 - c_3\alpha|x|)e^{-\alpha|x|}, \tag{10}$$

where c_0, c_1, c_2, and c_3 are scaling factors defined as:

$$c_0 = \frac{(1-e^{-\alpha})^2}{1 + 2e^{-\alpha}\alpha - e^{-2\alpha}}, \tag{11}$$

$$c_1 = \frac{-(1-e^{-\alpha^3})}{2\alpha^2 e^{-\alpha}(1+e^{-\alpha})}, \tag{12}$$

$$c_2 = \frac{-2(1-e^{-\alpha^4})}{1 + 2e^{-\alpha} - 2e^{-3\alpha} - e^{-4\alpha}}, \text{ and} \tag{13}$$

$$c_3 = \frac{(1-e^{-2\alpha})}{2\alpha e^{-\alpha}}, \tag{14}$$

with α a smoothing term. Monga et al. [Mon92] noted that smaller values for α are often required when estimating second derivatives compared to when estimating first derivatives.

Aside from one set of supplemental experiments at the end of Section 4, we have set $\alpha = 1.0$, since that same value for α was used in a prior work ([Hau21]) that explored the **DF** method.

2.3 Aims and Motivations

This work is motivated by prior work that considered accuracy and computational performance of hypersurface curvature computation [Hau19].

Energy-efficient algorithm strategies for linear system solutions has been one focus of prior work. Köhler and Saak [Kö19], for example, have explored such strategies, including: (1) combining certain evaluation activities (to save on data transfers); (2) use of a Newton-type method to compute matrix sign; (3) use of Gauss-Jordan elimination in lieu of LU-decomposition; and (4) changing one internal storage scheme to improve cache utilization.

Our strategies here, described later, also include efforts aimed at saving on data transfers and improving cache utilization, including in linear system components of hypersurface curvature determination.

3 METHODS/STRATEGIES

The strategies to achieve energy efficiency in the hypersurface curvature computation, primarily involving improvements in linear algebraic computations and compiler products, are described in this section.

3.1 Linear Algebra Memory & Computational Improvements

Hypersurface curvature determination in volumetric datasets using the methods previously reported in the literature requires estimation of derivative quantities in the dataset followed by steps including finding inverses, determinants, and performing matrix multiplication. These operations implement the equations described above. Standard solver libraries can be used

to compute those, and we initially used the *Armadillo* library [San16] for that. (Later, we report comparisons versus use of Armadillo.) Armadillo has several characteristics that are not energy-optimal, though, for application here. For one, it uses its own internal format for matrices, which incurs some overhead in both time and energy consumption. Also, our core matrices are symmetric (but not necessarily positive definite). (N.B.: the matrix product is not symmetric, though.) Inspection of the Armadillo library functions revealed that the inverse and determinant computation in them did not exploit matrix symmetry for (non-positive definite) symmetric matrices, which means that there are redundant computations in the case of such matrices. Our approach avoids these redundancies, thereby saving both energy and time.

Our approaches to improve energy efficiency of linear algebraic-based computations for hypersurface curvature determination involved replacement of Armadillo with our own realizations that store matrices as standard arrays and avoid unnecessary computations performed in the Armadillo code. Some aspects of the savings our realizations achieve for matrix determinant follow. 3×3 matrices of the form:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{15}$$

have determinants of the form:

$$a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}. \tag{16}$$

Computing such a determinant involves loading the 9 matrix entries into the control unit of the CPU and performing 5 additions and 9 multiplies. Ultimately, Armadillo realizes these same actions. Our linear algebra strategy for determinant calculation, however, exploits that in our symmetric matrices, $d = b$, $g = c$, and $h = f$. Thus, for our hypersurface curvature computation the determinant of the matrix shown in Eq.15 has the form:

$$a \begin{vmatrix} e & f \\ f & i \end{vmatrix} - b \begin{vmatrix} b & f \\ c & i \end{vmatrix} + c \begin{vmatrix} b & e \\ c & f \end{vmatrix}. \tag{17}$$

The Eq.17 can be reduced to a form such as:

$$(ei - ff)(a) - (bi - \mathcal{K})(b) - ecc, \tag{18}$$

where $\mathcal{K} = 2cf$, with $\mathcal{K}$ computed in our algorithm as a two step process:
$\mathcal{K} = cf$; $\mathcal{K} = \mathcal{K} + \mathcal{K}$.

The net effect of our approach to finding the determinant is a computation with 5 additions and only 8 multiplies that also only loads 6 matrix entries into the control unit of the CPU. Our approach thus reduces loading, reduces register pressure, and can remove one instance of the most expensive operation in the process.

This arithmetic approach extends comparably to the computation of the inverse.

Our linear algebra strategy thus has four components in all. They include: (1) avoiding special, generic matrix storage formats (and unnecessary storage of those) via use of standard matrix storage; (2) exploiting matrix symmetry and size in determinant computation; (3) exploiting matrix symmetry and size in finding matrix inverses; and (4) exploiting knowledge of matrix multiplicand form in finding matrix products.

3.2 Compilation Improvements

We also considered and report here on two categories of compilation-based approaches for energy reduction for the language environment we used, C/C++. C/C++-based solutions are known to be among the most energy-efficient [Per17]. The two approaches are (1) the use of a very high optimization compiler setting (our compiler was gcc, and its very high optimization setting is the O3 one), denoted VH henceforth, and (2) the use of the special mathematical operation optimization setting (in gcc, this setting is called *ffastmath*), denoted FM henceforth. (N.B., because FM relaxes some floating point compliance within the compiler, it can impact accuracy. In our experiments on a real dataset, the average change in the curvature values due to the use of FM was 2.8×10^{-16} and the maximum change in any of the curvature values due to the use of FM was 1.5×10^{-14}.)

(Automated loop unrolling, denoted Unroll henceforth, was also considered and is reported later. It cannot be considered a general improvement in application to methods here, as discussed later.)

3.3 B-Spline Library-Related Improvements

For the **BS** method, we implemented two variants of the method. Our first variant, simply denoted **BS**, utilized the header-only *vspline* library [kfj23]. The second variant, denoted **BSWM5**, used the B-Spline functionality present in the larger *WildMagic 5* library [Ebe04] Our results, presented later, consider the energy usage of both of these variants of the B-Spline-based approach.

4 RESULTS

Next, results are reported. All results were determined on a CPU similar to one used in some related work. Here, the environment used one core of an Intel Core i5-8279U CPU. The operating system used was Ubuntu Server 22.04.1 GNU/Linux. All runs were built using the gcc compiler (version 11.3.0). (N.B., we found that code compiled with gcc was consistently among the best (in terms of energy usage) compared to both

clang and icc.) Energy measurements and timings were done via the Performance API (PAPI) software [Bro00], which in turn measured energy via the processor's Running Average Power Limit (RAPL) feature. RAPL, which is supported on Sandy Bridge and newer Intel CPUs [Kha18], allows measuring CPU power usage. RAPL uses hardware counters in conjunction with factors such as temperature and leakage to estimate CPU energy usage, and it supports measurement of energy usage using different domains including *package* (total CPU package energy usage), *PP0* (CPU core energy usage), and *DRAM* (DRAM controller). A previous study found RAPL measurements to be accurate while also exhibiting negligible overhead [Kha18]. The governor on the computer was set to the "performance" mode in order to maintain more consistent clock rates, and all code was run as the "root" user (in order to ensure access to the RAPL hardware) using the *taskset* utility (in order to ensure that all runs were done on the same processor core).

To find the cost of determining hypersurface curvature via canned library (i.e., Armadillo) based linear algebraic routines, we performed timing and energy use. For all experiments, five runs were taken and the trimmed mean (i.e., discarding most and least energy usage run) was computed. For one experiment, a $256\times256\times256$ volumetric dataset mathematically generated from a polynomial function used in other reported work on curvature in volumes and called the "Genus3" polynomial there ([Hau20], Eqn. 39) was used. On such a dataset, the baseline (optimized) result (i.e., compilation using gcc with -O2 on the system reported above) for hypersurface curvature determination using the OP method was found to use 776.9 J (for CPU package energy) and take 46.6 seconds of CPU time. Using the VH optimization improved energy use by a factor of 1.066 and run-time by a factor of 1.075. Coupling VH and FM resulted in total improvement factors of 1.077 and 1.082, respectively. However, replacement of Armadillo with our linear algebraic strategies coupled with the VH compilation strategy resulted in both energy and time improvement factors of 1.25. (N.B., the measured energy usage and time typically varied by about 1% from run to run.)

A breakdown of individual linear algebraic-related strategy time and (package) energy improvement factors are shown in Table 1 (versus original baseline ("Base") energy use). To ensure that these experiments consider the overall impact of the underlying libraries in a wide cross-section of scenarios, we computed these results on a randomly-generated 256^3 volume, without regard to the end curvature estimator used. (Of note here: time savings does not always equal energy savings.) Total energy and time improvements from these strategies taken together are also shown.

Strategy	Base Energy	Energy Imp.	Time Imp.
Storage	25.40J	11.98x	7.08x
Det.	6.26J	2.95x	2.54x
Inv.	11.27J	1.43x	1.37x
Mult.	11.12J	1.40x	1.49x
Total:	**54.05J**	**2.70x**	**2.18x**

Table 1: L.A. Strategy Energy Use (J) & Improvement

The numbers in Table 1 show the energy usage measurements of these operations in isolation (i.e., not when used in combination as part of a larger problem solving task). When these operations are performed in combination as part of larger task, such as curvature determination, the improvements can be even more substantial. In fact, the results in Table 1 account for less than one-half the total net improvement in time and energy. Such results for these optimizations incorporated into curvature determination are described next.

Table 2 shows overall energy usage results for the complete curvature determination of the four methods when applied to a $256\times256\times72$ brain magnetic resonance angiography (MRA) dataset. The BS Method library-related results are compared in the last two columns of the table. The entries marked "Custom" here refer to the use of our linear algebraic strategies rather than use of the Armadillo library. The two best choices here are (1) VH with FM and the linear algebraic strategies and (2) VH with FM and automated loop unrolling with the linear algebraic strategies. Average energy consumption improvement is 16.72% for the first choice, which is equivalent to 1.20 times improvement. For the second choice, average energy consumption improvement is 15.91%, which is equivalent to 1.19 times improvement. Since the automated loop unrolling results are, overall, yielding performance improvement about the same as optimization without them, we recommend the first choice here, however practitioners using the DF curvature determination method would still benefit slightly from its use, it appears.

Table 3 shows run times for the same dataset. The locations of the red cells differs between Table 2 and Table 3, which is indicative of something that has also been observed in prior studies: there is not always a direct correspondence between run time and energy usage. Thus, as a general conclusion, results here (again) indicate that optimization for energy consumption and optimization for run-time may require different methods and approaches.

Fig. 1 shows a composite rendering of three maximum intensity projections (MIP), one per principal curvature, of the MRA dataset. (κ_1's MIP forms the red channel of the composite here. κ_2's forms the green channel. κ_3's, which was clamped to remove low values, forms the blue channel.) MIP is popular for raw MRA data

Method	OP	TE	DF	BS	BSWM5
Baseline (Armadillo)	176.24 J	135.58 J	185.28 J	230.81 J	399.73 J
Baseline (Custom)	-11.91%	-17.75%	-8.49%	-10.24%	-6.52%
Baseline FM (Armadillo)	-0.84%	-0.92%	3.91%	-0.51%	-0.35%
Baseline FM (Custom)	-12.87%	-17.56%	-12.19%	-11.08%	-10.65%
Baseline Unroll (Armadillo)	2.42%	0.03%	-1.27%	4.86%	-5.49%
Baseline Unroll (Custom)	-15.02%	-17.93%	-10.86%	-5.35%	-12.34%
Baseline Unroll FM (Armadillo)	2.75%	-0.61%	-2.46%	3.11%	0.67%
Baseline Unroll FM (Custom)	-15.35%	-18.96%	-9.65%	-4.45%	-10.41%
VH (Armadillo)	-4.17%	-3.49%	-4.45%	-0.99%	-6.42%
VH (Custom)	-16.86%	-21.10%	-17.21%	-10.41%	-10.57%
VH FM (Armadillo)	-4.86%	-4.54%	-6.97%	-3.38%	-3.64%
VH FM (Custom)	**-17.57%**	**-21.55%**	-17.53%	**-12.78%**	-14.17%
VH Unroll (Armadillo)	-1.21%	-3.39%	-4.09%	0.64%	0.63%
VH Unroll (Custom)	-16.14%	-20.37%	-16.16%	-10.37%	-11.94%
VH Unroll FM (Armadillo)	-2.86%	-3.61%	-4.68%	-0.16%	-5.91%
VH Unroll FM (Custom)	-16.75%	-20.62%	**-17.80%**	-9.91%	**-14.47%**

Table 2: Energy usage on the MRA dataset (trimmed means of five runs), relative to Baseline for each curvature determination method in conjunction with the optimization strategies. Cell backgrounds are color mapped based on energy usage relative to baseline. In each column, the **bold** entry indicates the approach with lowest energy usage for that method and the yellow bordered entry indicates the approach with the fastest run time for that method. Overall, best energy improvement averages about 1.20 times.

Method	OP	TE	DF	BS	BSWM5
Baseline (Armadillo)	10.65 s	7.85 s	11.43 s	13.00 s	25.91 s
Baseline (Custom)	-11.12%	-18.02%	-8.54%	-10.45%	-5.24%
Baseline FM (Armadillo)	-0.42%	-0.65%	3.53%	-0.55%	-0.47%
Baseline FM (Custom)	-12.02%	-17.85%	-11.80%	**-11.49%**	-3.86%
Baseline Unroll (Armadillo)	2.45%	-0.24%	-1.63%	4.66%	-0.23%
Baseline Unroll (Custom)	-14.15%	-18.33%	-11.03%	-5.69%	-6.21%
Baseline Unroll FM (Armadillo)	3.15%	-1.08%	-2.31%	3.10%	0.81%
Baseline Unroll FM (Custom)	-14.40%	-19.12%	-9.90%	-4.88%	-3.70%
VH (Armadillo)	-3.92%	-3.94%	-5.05%	0.91%	-0.91%
VH (Custom)	-15.93%	-21.45%	-17.29%	-8.08%	-3.69%
VH FM (Armadillo)	-4.69%	-4.76%	-7.91%	-1.43%	1.93%
VH FM (Custom)	**-16.62%**	**-21.96%**	-18.05%	-10.22%	-7.35%
VH Unroll (Armadillo)	-1.13%	-4.01%	-4.97%	3.85%	0.70%
VH Unroll (Custom)	-15.46%	-21.10%	-16.86%	-6.78%	-6.15%
VH Unroll FM (Armadillo)	-3.29%	-4.14%	-5.75%	2.89%	-0.86%
VH Unroll FM (Custom)	-16.21%	-21.35%	**-18.22%**	-7.12%	**-7.61%**

Table 3: Run times on the MRA dataset (trimmed means of five runs), relative to Baseline for each curvature determination method in conjunction with the optimization strategies. Cell backgrounds are color mapped based on energy usage relative to baseline. In each column, the **bold** entry indicates the approach with lowest run time for that method.

rendering. To our knowledge, this figure here is the first MIP rendering of hypersurface curvature features in MRA data, however. **OP**'s results were used to produce Fig. 1. The amount of energy used to produce the rendering is 146.94 J on average (146.5 J for curvature computation and 0.42 J for MIPs).

Table 4 presents the results of energy usage experiments for two other sets of parameters for both **OP** and **DF** versus the values used in the prior experiments. In this table, **OP** N denotes application of **OP** with an estimation filter of size N and **DF** α denotes application of **DF** with a smoothing parameter of value α. As N increases, the energy usage of **OP** increases. This is expected, as larger N values require more computation. Despite **OP** 5 and **OP** 9 being 28% smaller and larger, respectively, than **OP** 7, the energy differences are substantially less than that. To some extent, this is to be expected, as only some of the computation is related to the size of the estimation filter. **OP** 5 uses about 0.95 times the energy of **OP** 7 and **OP** 9 uses about 1.07 times the energy of **OP** 7, with energy usage scaling up more slowly when all strategies are used. The energy differences are much less noteworthy on **DF**. To some extent, this is to be expected, as α does not substantially change the amount of computation. We do note that running **DF** with $\alpha = 0.5$ uses slightly less energy than running it

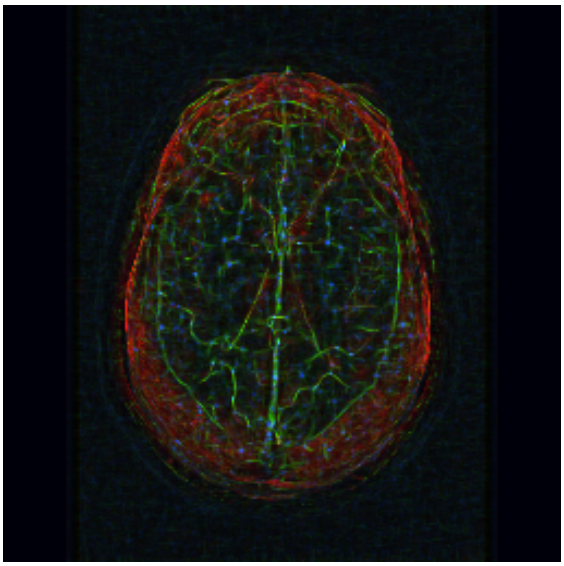

Figure 1: MIP: κ_1, κ_2, κ_3 Composite, MRA Dataset.

with $\alpha = 1.0$ and $\alpha = 2.0$. Our hypothesized reason for this is that the optimal filters used by **DF** rely on the *pow* function, which can be more optimal for certain exponents, in particular ones like 0.5 and 2, which appear to be exhibited more by the $\alpha = 0.5$ than by the $\alpha = 1.0$ and $\alpha = 2.0$ settings.

5 CONCLUSIONS

Most of our strategies exploit known properties of the hypersurface curvature determination domain to reduce energy consumption (and computation time!). Our storage-based strategy yielded the largest energy (and time) improvement for the linear algebraic computational components of the hypersurface curvature computation, achieving nearly a 12-fold improvement in energy usage, when considered in isolation. Overall, the linear algebraic computational strategies achieved nearly a 3-fold improvement in energy usage for those components of the hypersurface curvature determination, when taken together.

For practitioners using a B-spline based method for hypersurface curvature determination, use of the *vspline* library appears to enable significantly lower energy use than use of the *WildMagic 5* library; that results in about a 1.73 times improvement in energy consumption.

For practitioners, the best generic advice for energy-efficient hypersurface curvature determination is to use the VH and FM compilation strategies coupled with the linear algebraic strategies, which can result in about a 1.20 times energy improvement.

Our strategies can provide means for reduced energy consumption for hypersurface-based volume data analyses and visualizations. Data analyses and visualizations using them can thus extend battery life as well as reduce their environmental impact.

ACKNOWLEDGMENTS

The research reported here was conducted independently; it was not sponsored by any sort of energy-related organization.

6 REFERENCES

[Ald14] G. Aldrich, A. Gimenez, M. Oskin, R. Strelitz, J. Woodring, L. H. Kellogg, and B. Hamann. Curvature-based crease surfaces for wave visualization. *Vision, Modeling & Vis.*, pp. 39–46, 2014.

[BM21] J. Beckitt-Marshall. *Improving Energy Efficiency through Compiler Optimizations*. Bowdoin College (Honors Project 239 Report), 2021.

[Bra10] L. Bravo and M. Aldana. Volume curvature attributes to identify subtle faults and fractures in carbonate reservoirs: Cimarrona formation, middle magdalena valley basin, colombia. *2010 SEG Annual Meeting*. OnePetro, 2010.

[Bro00] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *Int'l J. High Perf. Comp. Appl.*, 14(3):189–204, 2000.

[Der90] R. Deriche. Fast algorithms for low-level vision. *IEEE T-PAMI*, 12(1):78–87, Jan 1990.

[Ebe04] D. Eberly. *3D game engine architecture: engineering real-time applications with Wild Magic*. CRC Press, 2004.

[EM20] D. El Mezeni and L. Saranovac. Fast guided filter for power-efficient real-time 1080p streaming video processing. *J. Real-Time Image Processing*, pp. 511–525, 2020.

[Fly89] P. Flynn and A. Jain. On reliable curvature estimation. *Proc., IEEE Comput. Vision and Pat. Recog. '89*, pp. 110–116, 1989.

[Fuc20] H. Fuchs, A. Shehabi, M. Ganeshalingam, L.-B. Desroches, B. Lim, K. Roth, and A. Taso. Comparing Datasets of Volume Servers to Illuminate their Energy use in Data Centers. *Energy Efficiency*, 13:379–392, 2020.

[Gup22] U. Gupta, Y. Kim, S. Lee, J. Tse, S. Hsien-Hsin, G.-Y. Wie, D. Brooks, and C.-J. Wu. Chasing carbon: The elusive environmental footprint of computing. *IEEE Micro*, 42(4):37–47, 2022.

[Hau19] J. D. Hauenstein and T. S. Newman. Exhibition and evaluation of two schemes for determining hypersurface curvature in volumetric data. *J. WSCG*, 27(2):121–129, 2019.

Method	OP 9	OP 7	OP 5	DF 2.0	DF 1.0	DF 0.5
Baseline (Armadillo)	188.44 J	175.86 J	166.03 J	189.02 J	186.74 J	185.25 J
VH Unroll FM (Custom)	154.15 J	146.66 J	140.73 J	155.11 J	152.78 J	151.05 J

Table 4: Energy usage on the MRA dataset (trimmed means of five runs) for Baseline vs. all strategies at three different parameter settings for **OP** and **DF**.

[Hau20] J. D. Hauenstein and T. S. Newman. Descriptions and evaluations of methods for determining surface curvature in volumetric data. *Computers & Graphics*, 86:52–70, 2020.

[Hau21] J. D. Hauenstein and T. S. Newman. New methods and novel framework for hypersurface curvature determination and analysis. *J. WSCG*, 29(1–2):11–20, 2021.

[Hen20] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *J. Machine Learn. Res.*, 21:1–43, 2020.

[Hir01] Y. Hirano, A. Shimizu, J.-i. Hasegawa, and J.-i. Toriwaki. A tracking algorithm for extracting ridge lines in three-dimensional gray images using curvature of four-dimensional hypersurface. *Systems and Comput. in Japan*, 32(12):25–37, 2001.

[Hir18] Y. Hirano. Categorization of lung tumors into benign/malignant, solid/GGO, and typical benign/others. K. Suzuki and Y. Chen, editors, *Artificial Intelligence in Decision Support Systems for Diagnosis in Medical Imaging*, pp. 193–208. Springer Nature, 2018.

[kfj23] kfj. vspline: C++ template library for B-splines, 2023. https://bitbucket.org/kfj/vspline/.

[Kha18] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou. Rapl in action: Experiences in using rapl for power measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 3(2), mar 2018.

[Kin03] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: methods and applications. *Proc., IEEE Visualization '03*, pp. 513–520, 2003.

[Kö19] M. Köhler and J. Saak. Frequency Scaling and Energy Efficiency regarding the Gauss-Jordan Elimination scheme with Application to the Matrix-sign-Function on OpenPOWER 8. *Concurrency and Computation: Practice and Experience*, 31(6):e4504, 2019.

[Lan21] L. Lannelongue, J. Grealey, and M. Inouye. Green Alogorithms: Quantifying the Carbon Footprint of Computation. *Advanced Science*, 8:202100707, 2021.

[Lan23] L. Lannelongue and M. Inouye. Carbon Footprint Estimation for Computational Research. *Nature Reviews Methods Primers*, 3:Article 9, 2023.

[Lin19] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. *Proc., IEEE/CVF Int'l Conf. on Computer Vision '19*, pp. 7083–7093, 2019.

[Mon92] O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surface features. Research Report RR-1599, INRIA, 1992.

[Pap07] L. Papaleo. An approach to surface reconstruction using uncertain data. *Int'l J. of Image and Graphics*, 07(01):177–194, 2007.

[Per17] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. Fernandes, and J. Saraiva. Energy efficiency across programming languages. *Proc., ACM SIGPLAN Int'l Conf. Soft. Lang. Engg.*, pp. 256–267, 2017.

[San16] C. Sanderson and R. Curtin. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

[Suz18] H. Suzuki, Y. Kawata, N. Niki, T. Sugiura, N. Tanabe, M. Kusumoto, K. Eguchi, and M. Kaneko. Automated assessment of aortic and main pulmonary arterial diameters using model-based blood vessel segmentation for predicting chronic thromboembolic pulmonary hypertension in low-dose ct lung screening. *Medical Imaging 2018: Comput.-Aided Diagnosis*, volume 10575, 2018.

[Tit15] J. Tithi, P. Ganapathi, A. Talati, S. Aggarwal, and R. Chowdhury. High-performance energy-efficient recursive dynamic programming with matrix-multiplication-like flexible kernels. *Proc., 29th Int'l Par. and Dist. Processing Symp.*, pp. 303–312, 2015.