Image-based Visualization of Large Volumetric Data Using Moments Supplemental Document

Tobias Rapp*
Karlsruhe Institute of Technology

Christoph Peters†
Karlsruhe Institute of Technology

Carsten Dachsbacher‡
Karlsruhe Institute of Technology

A Quantization and Compression

The algorithm that computes a quantization curve is shown in algorithm 1. The algorithm determines an error threshold from quantizing the moment at index 1 with b_1 number of bits. The number of bits is then decreased as much as possible for each subsequent index l as long as the error stays below the threshold.

To evaluate the impact of our coding and quantization scheme, we compare it to the baseline of directly quantizing the bounded trigonometric moments $c_0 \in [0,1]$ and $c_j \in [-\frac{1}{\pi}, \frac{1}{\pi}]$ with a fixed number of bits. Afterwards we perform lossless compression. Fig. A.1 compares this baseline approach to our coding scheme. We vary the amount of quantization and compare the resulting image size and accuracy of reconstructed scalar values. Our novel coding scheme achieves a reduction in size as well as an increase in accuracy for all datasets.

Algorithm 1: Computing a quantization curve

```
Data: Moment image M
Input  : Initial bits b_1
Output : Quantization table t
t[0] ← 16
t[1] ← b_1
// Determine error threshold
M′ ← quantize(M, 1, b_1)
e_T ← rRMSE(moments(M), moments(M′))
for l ← 2 to m do
    t[l] ← t[l − 1]
    // Reduce number of bits until we reach the threshold
    for b ∈ {t[l − 1], ..., 1} do
        M′ ← quantize(M, l, b) // Quantize index l with b
        e ← rRMSE(moments(M), moments(M′))
        if e ≥ e_T then
            break
        t[l] ← b
```

B Number of Moments

Fig. B.1 illustrates the number of moments in each pixel of the three moment images. The amount of moments adapts to the complexity of the data. For example, in the turbine dataset, the turbine blades and swirling regions require more moments, whilst the surrounding volume requires fewer. Note that this adaptation is independent of the employed transfer function. For the Richtmyer-Meshkov and the Rayleigh-Taylor dataset, most pixels contain the maximum number of moments. This is a clear indication that we should use a higher maximal number of moments for these datasets. For the Richtmyer-Meshkov dataset, we have increased the maximal number of moments from 100 in Fig. B.2 (a), to 120 in (b) and to 150 in (c). The quality visibly improves in (b) and again in (c).

C Uncertainty Quantification

We visualize the 90th error percentile in Fig. C.1 for each pixel in moment images from all three datasets. This illustrates areas of little and high uncertainty. For example, the Richtmyer-Meshkov dataset in Fig. C.1 (a) shows a comparatively high error, due to its complexity. In contrast, the smoother turbine dataset in Fig. C.1 (c) shows little uncertainty. Regions near the turbine blades have a higher error than the rest of the image.

In the supplemental video, we further employ these error bounds to visualize uncertainty using temporal animation. Specifically, we choose and fix a value in $[-1,1]$ before rendering. At each step during ray marching, this value is scaled by the error bound for each pixel and added to the reconstructed scalar values. This leads to possible scalar densities with respect to the error bounds. Changing the scaling factor over time leads to flickering in uncertain areas which attracts attention.

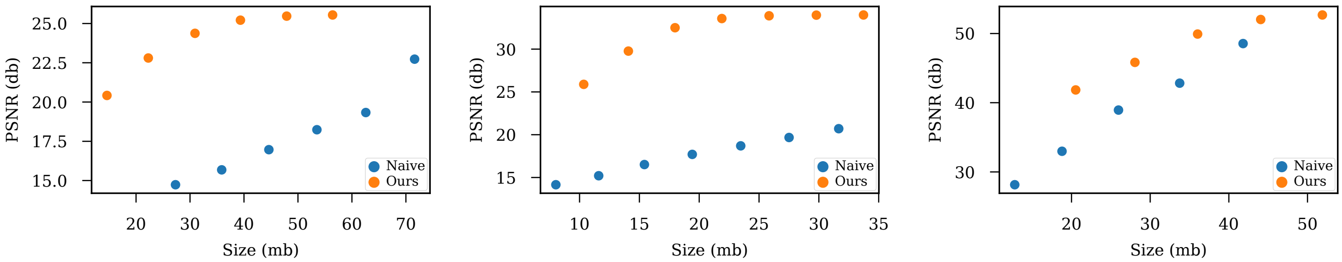
D Scalability

To investigate the scalability of our approach, we measure CPU run-times on the turbine dataset, which requires expensive SPH interpolation during ray marching. The results are shown in Fig. D.1 and are performed on an AMD Epyc Milan cloud instance with differing amounts of vCPU cores.

Reference volume rendering with ray marching is shown in (a) and scales nearly linearly with the number of vCPU cores. This is also visible from the speedups shown in (d). Generating a moment image (b) is approximately twice as slow compared to this reference ray marching. The run-time is dominated by the ray marching, which includes the computation of 100 moments for each pixel. But once generated, rendering a moment image (c) is an order of magnitude faster compared to the reference. When the moment image is already decoded and prepared for rendering, which has to be performed only once, rendering becomes nearly two orders of magnitude faster.

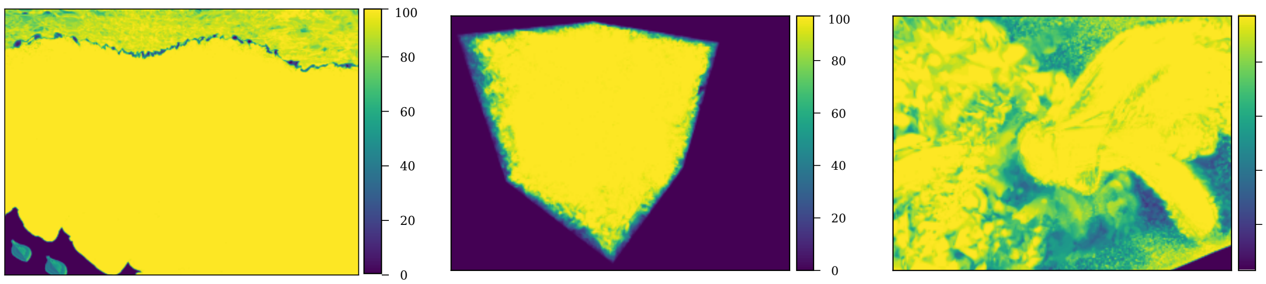
Generating and rendering moment images is well suited to be executed on massively parallel hardware, such as GPUs, since all steps are trivially parallelizable over the pixels in an image. An exception is the lossless encoding and decoding, which still benefits from CPU parallelization. The speedups in (d) thus show linear scaling for the moment image generation. For rendering, the ray marching step scales nearly linearly. The moment preparation, i.e. the inversion of the coding and the computation of Lagrange multipliers, does not scale well beyond 32 vCPU cores. This might be due to a memory bottleneck as this step is very memory intensive. Our GPU implementation relies on shared memory for these computations, which significantly reduces the run-time and improves scaling to a large amount of GPU cores.

*e-mail: tobias.rapp@kit.edu
†e-mail: christoph.peters@kit.edu
‡e-mail: dachsbacher@kit.edu



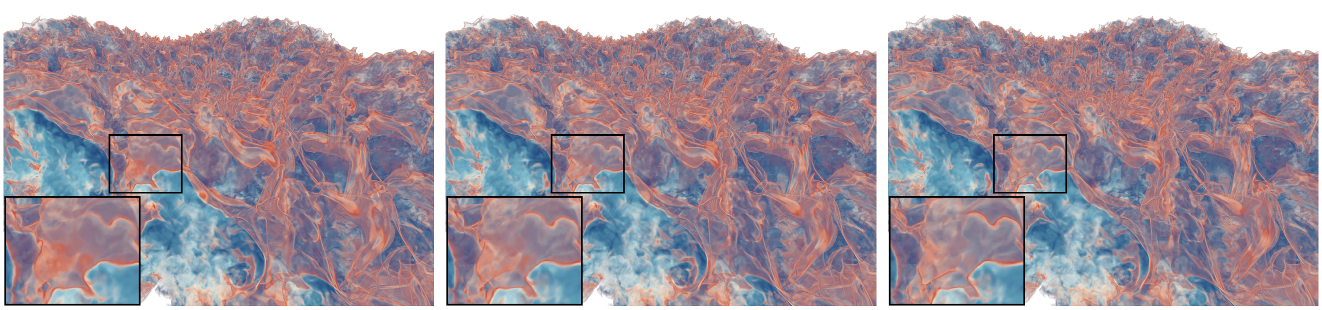
(a) Richtmyer-Meshkov (b) Rayleigh-Taylor (c) Turbine

Figure A.1: Comparison of our coding and compression scheme with a naive approach that directly quantizes and compresses the moments. Our approach leads to an increased accuracy of reconstructed scalar values at smaller image sizes.



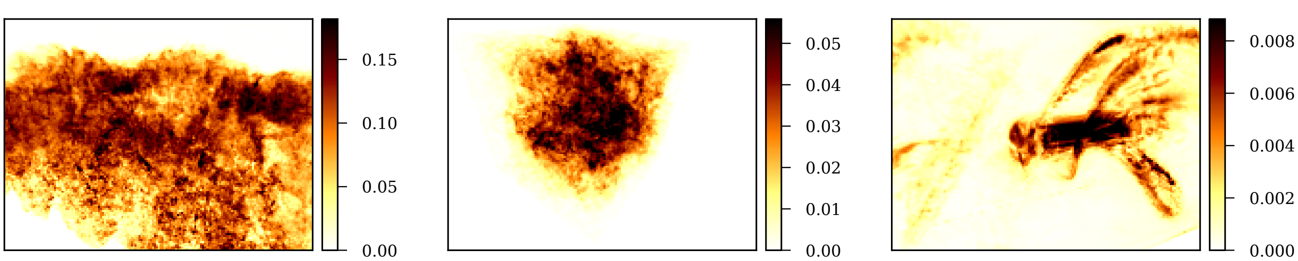
(a) Richtmyer-Meshkov (b) Rayleigh-Taylor (c) Turbine

Figure B.1: Number of moments per pixel for moment images created from our datasets. Images from the Richtmyer-Meshkov (a) and Rayleigh-Taylor (b) datasets mostly select the maximum of 100 moments. This indicates that a higher number of moments would improve the reconstruction. For the turbine (c), this applies to a few regions near swirling regions and the turbine blades.



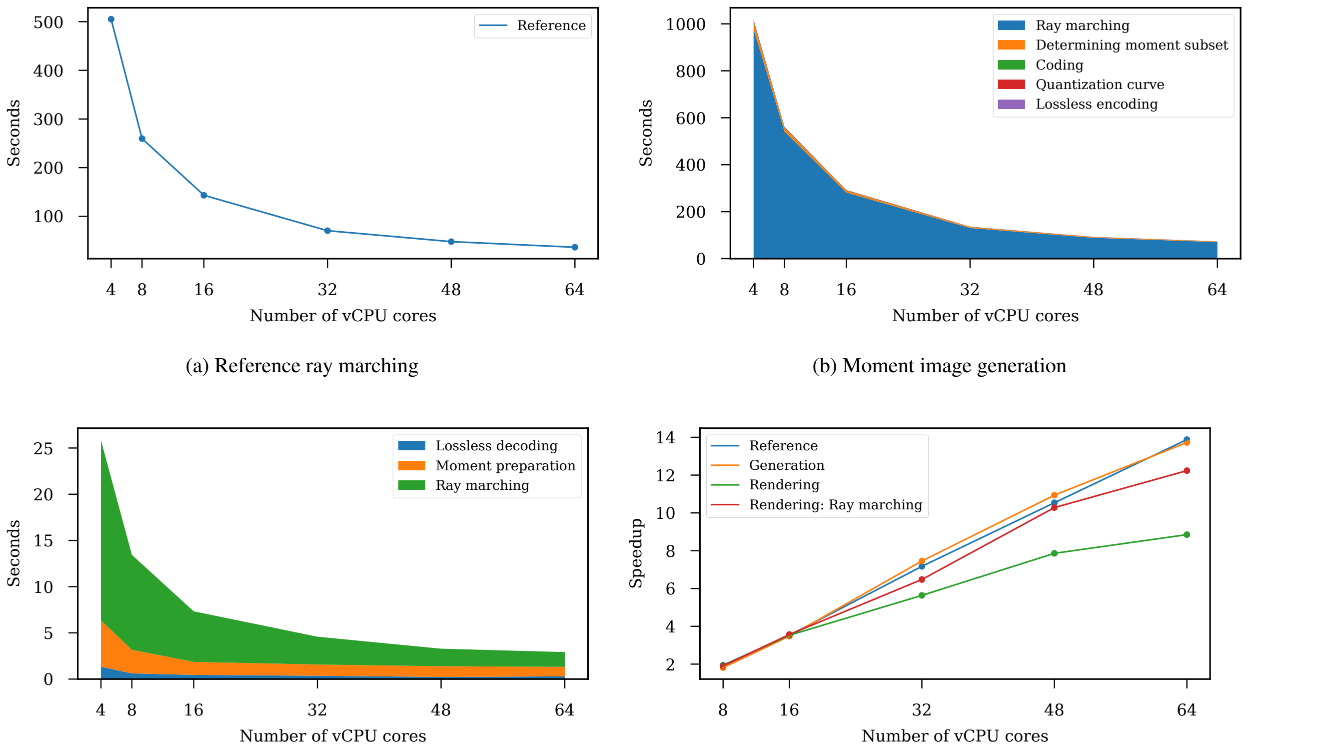
(a) Maximum of 100 moments (b) Maximum of 120 moments (c) Maximum of 150 moments

Figure B.2: Moment images of the Richtmyer-Meshkov dataset with a different maximal amount of moments. The accuracy visibly improves when more moments can be used.



(a) Richtmyer-Meshkov (b) Rayleigh-Taylor (c) Turbine

Figure C.1: For each pixel in moment images from our datasets, we compute the 90th percentile of the absolute differences from the reference to the reconstructured scalar values. The maximal number of moments is the same as in Fig. B.1.



(a) Reference ray marching

(b) Moment image generation

(c) Moment image rendering

(d) Speedup due to parallelization

Figure D.1: Run-time measurements of the turbine dataset with varying numbers of vCPU cores for reference ray marching (a), moment image generation (b), and moment image rendering (c). The speedup compared to using 4 vCPUs is shown in (d). For rendering, the lossless decoding and moment preparation steps have to be performed only once. Therefore, we also report the speedup of the ray marching step.