Is Smaller Always Better? - Evaluating Video Compression Techniques for Simulation Ensembles

Patrick Ruediger
Visual Information Analysis Group, Technische Universität Kaiserslautern, Germany

Christoph Garth
Scientific Visualization Group, Technische Universität Kaiserslautern, Germany

Hans Hagen
Computergraphics and HCI Group, Technische Universität Kaiserslautern, Germany

Heike Leitte
Visual Information Analysis Group, Technische Universität Kaiserslautern, Germany

Abstract

We provide an evaluation of the applicability of video compression techniques for compressing visualization image databases that are often used for in situ visualization. Considering relevant practical implementation aspects, we identify relevant compression parameters, and evaluate video compression for several test cases, involving several data sets and visualization methods; we use three different video codecs. To quantify the benefits and drawbacks of video compression, we employ metrics for image quality, compression rate, and performance. The experiments discussed provide insight into good choices of parameter values, working well in the considered cases.

2012 ACM Subject Classification Computing methodologies → Image compression; Applied computing → Physics; Applied computing → Engineering

Keywords and phrases Image Database, CinemaDB, Video Compression, Evaluation, Benchmark, In-situ

Digital Object Identifier 10.4230/OASIcs.iPMVM.2020.10

Funding This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 252408385 – IRTG 2057.

1 Introduction

Today's supercomputer architectures allow computational scientists to perform research with increasingly complex models producing high-resolution, high-fidelity data in increasingly shorter times. It is often no longer possible to store data set at full resolution, and it is also extremely inefficient and expensive to transfer such large data sets from tertiary storage into memory for analysis. Consequently, *in situ* visualization – i.e., the generation and storage of visualizations/images done as part of an ongoing computer simulation – has been developed and as a means to avoid most of the difficulties one would have to address when performing large-scale data visualization via traditional approaches [21, 6]. Nevertheless, while an in situ approach is advantageous in many ways it limits the possibilities of "complete data exploration" to the a set of pre-generated visualizations. Ahrens et al. [1, 23] recognized that generating a large set of images covering the underlying parameter value space well still allows a scientist to meaningfully analyze and discover important model behaviors while accelerating the exploratory process significantly as all visualizations already have been generated.

For complex visualizations generated, for example, by sampling large and/or high-dimensional visualization parameter spaces, the databases needed for storage can still be of substantial size. To reduce data size and thereby make possible a more efficient management

2nd International Conference of the DFG International Research Training Group 2057 – Physical Modeling for Virtual Manufacturing (iPMVM 2020).
Editors: Christoph Garth, Jan C. Aurich, Barbara Linke, Ralf Müller, Bahram Ravani, Gunther Weber, and Benjamin Kirsch; Article No. 10; pp. 10:1–10:18

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and processing of visualization databases, we adapt image and video compression techniques. The goal is to leverage inter-image similarity typically encountered in video image frame sequences and used to achieve substantial compression. By "linearizing" a visualization image database it is possible to adapt video compression in support of much more efficient data analysis. This approach was first investigated by Berres et al. [7], where it was shown that approach is feasible and beneficial. However, their linearization approach and choice of compression parameter values, e.g., video codec and image encoding, were ad hoc. Our work is motivated by the goal of gaining more insight into the effects of these parameters concerning overall usefulness of video-compressed image databases.

We present the results of a broader investigation of the different aspects of applying video compression to visualization image databases. Specifically, we consider the following issues:

- We quantitatively evaluate video compression for four different data sets with significantly different image characteristics that are expected to affect compression. We specifically consider streamline and isosurface visualizations. We employ three commodity, general-purpose and easy-to-use video codecs (H.264 [26], H.265 [11], VP9 [15]). We consider several quality metrics and compression/de-compression efficiency in relationship to compression ratio, serving as a central parameter for all three codecs.
- Extending the research of Berres et al. [7], who considered the compression of RGB images, we take into account implementation issues of visualization image databases. Specifically, we compress scalar value and depth images; perform composition of visualization images; empirically demonstrate that video compression can be applied successfully to visualization images.
- We investigate the effects of linearization of a high-dimensional visualization image space on compression efficiency and show that ordering matters and a "good order" can significantly improve the final compression rate.

Based on our prior work [4], we suggest parameters suitable for general-purpose use.

After discussing related work in Section 2, we describe the used experimental setup, i.e., the compression pipeline (Section 3) and its major parameters, followed by a description of the metrics used for evaluation 4. Quantitative results are provided and analyzed in Section 5, and we discuss practical recommendations. Finally, we address limitations of our study and provide conclusions in Section 6.

2 Related Work

There is a rich body of work focusing on data reduction for computational model output. Among the typical approaches are multi-resolution techniques, adaptive refinement, compression; a survey and overview of techniques typically utilized in situ is given by Li et al. [18].

Common to all these techniques is that only reduced data is available for post hoc analysis and visualization. Often, explicit error bounds are not available, or very difficult to estimate except in specific circumstances (e.g. [16]). Thus, accuracy of analysis is in direct competition with data reduction, where it is essential to hit the sweet spot between reduction rate and data quality [3].

Taking an alternate approach, in situ visualization [21, 6] generates visualization imagery on full-fidelity data while it resides in memory, both eschewing storage costs as well as producing accurate images, but in turn limiting the flexibility of exploration. The Cinema approach [1] generates a browsable, parameter-dependent database of visualization images that facilitates post hoc exploration in a manner that is satisfactory for many applications (e.g., [23]). Through clever implementation, such as storing scalar value images and depth

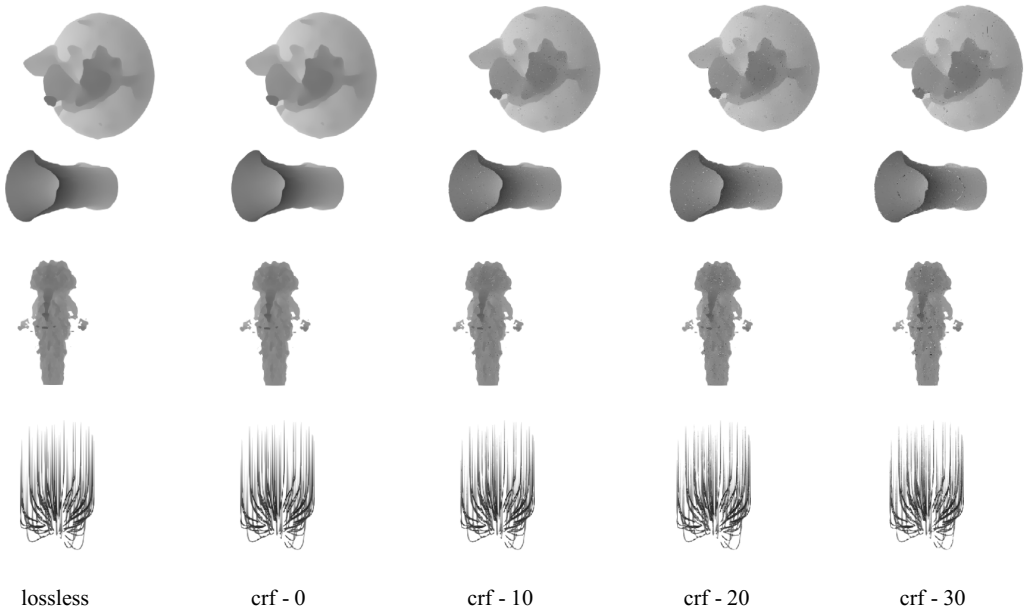


Figure 1 Depth Images before and after compression with the H.264 codec. The visually detectable differences are very small. Starting at crf= 10 the noise in the images is increasing with each step. Still most of the topological structures remain intact.

images instead of RGB images, compositing and color mapping can be employed to keep the image database size small. Lukasczyk et al. showed that under certain conditions it is possible to reconstruct parameter sets not stored in the database, such as e.g. camera positions, further enabling free exploration [20]. However, for very large parameter spaces that arise when combining many different visualizations in the interest of exploration, the visualization image database can become very large, again making its storage and use difficult. For specific scenarios, optimized image formats can be defined, e.g. in the case of volume rendering [28] or contour tree analysis [8]. However, this only marginally reduces the size of corresponding image databases. Image database storage also typically utilizes image compression techniques such as e.g. wavelet compression [30] or commodity image compression codecs (e.g. JPEG) in both lossy and lossless modes. However, compressing each image individually cannot leverage the high degree of similarity between images corresponding to closely neighboring parameter settings.

Concerning the delivery of in situ-generated visualization images, Ellsworth et.al.[12], Kageyema et.al.[17], and Biedert et al. [9] demonstrated the effectiveness of video compression to stream high-fidelity images rendered on a supercomputer to a desktop client. Applying this insight to visualization image databases, Berres et.al. [7] showed that visualization image databases are suitable for compression with video codecs, observing compression rates between 14% and 25%. Earlier, Sohn et.al.[25] proposed the idea of a specifically designed video compressor for volumetric images, using wavelet transformation and a temporal encoding to make efficient use of empty spaces. While they are maintaining good retrieval performances, the compression rates of 0.29% - 0.68% they achieve show their technique to be less effective.

Based on these prior works, and especially the proof-of-concept work of Berres et al. [7], we assume that video compression is a suitable way for reducing the size of large visualization image data bases. However, it is unclear which factors affect compression rate (i.e., data

reduction), image quality, and retrieval performance, which are the key pertinent aspects to consider when using compression for visualization image databases. In this paper, we investigate these aspects in a more comprehensive quantitative experiment.

When considering lossy compression, image quality is balanced against size reduction. In an excellent visual evaluation study, Turton et al. [4] showed that good compression rates can be achieved while preserving image quality, and found that the *structural similarity metric* [29] (SSIM) to be a good general indicator for image quality when comparing lossy compression to ground truth. In addition to other difference metrics such as PSRN, we hence base our evaluation heavily on the SSI metric.

3 Experimental Setup

In the following, we describe the pipeline we employ to carry out our study. A conceptual overview is given in Figure 2. To generate the visualization image database, we use Para-

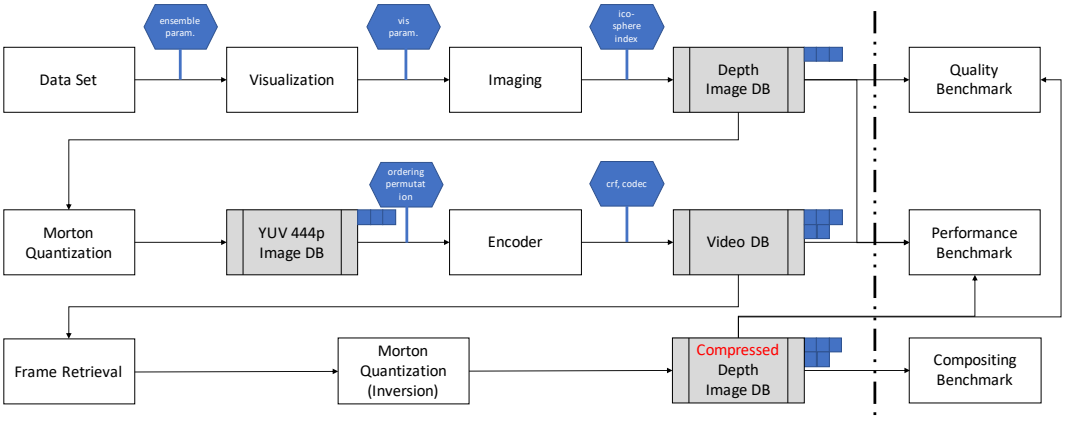


Figure 2 Overview of the image database generation, compression, and decompression pipeline underlying our study.

View [2] in combination with the TTK framework [27]. Prototype visualization pipelines are created in ParaView and stored as state files, which we then parameterize. The filters `ttkCinemaImaging` and `ttkCinemaWriter` are then used to render the images for a Cartesian product of parameter samples; including simulation time, camera positions, and isovalue (where appropriate). Camera positions are located on vertices of a spherical grid, and the cameras are aimed towards the data sets' centers.
Following the reasoning of Ahrens et al. [1], we focus on a modern implementation of visualization image databases based mainly on storing depth images instead of color images, as the former can be composited to combine different visualizations, avoiding a combinatorial size of combined visualization image databases. This allows us further to investigate the effects of compression on compositing as an additional benchmark. While we do not focus on scalar value images (used to achieve color mapping post hoc), our results (in particular, error metrics) apply directly to this class of images as well.

As an intermediate product, we obtain a depth image visualization database, which associates a depth image with each parameter set and represents the ground truth for error measurements. Image resolution is chosen as 512x512 throughout the entire study.

3.1 Data Sets and Visualizations

In the interest of covering a wide range of practical scenarios, we consider typical visualizations for four different data sets with very different characteristics.

3.1.1 Viscous Fingering

The data set is the result from a simulation of salt dissolving in water. The domain consists of a cylindrical flow, at the top of the cylinder a solid body of salt is placed that is dissolved by the water. The resulting fields are the velocity of the flow and concentration of salt in the water. The data set is time varying and consists of multiple parameter settings.

3.1.2 Geodynamo

This data set is the result from a numerical simulation of the earth's liquid outer core based on the effect of the geodynamo. The resulting domain is covering the whole outer core and is stored in a spherical unstructured grid. The resulting fields like pressure, temperature, velocity and magnetic field are characterized by high turbulence and large structures. The main challenge in this data set is its spherical domain and structures which are mostly hard to deal with standard approaches for cubic domains. In addition the time steps are in the order of thousands of years.

3.1.3 Jet Flow

The jet flow is an artificial unsteady flow simulation resembling the outlet of a jet engine. It is a well studied example data set for flow visualization and analysis and has very characteristic features. The resulting fields are the velocity and the temperature.

3.1.4 Metal Sheet Deformation

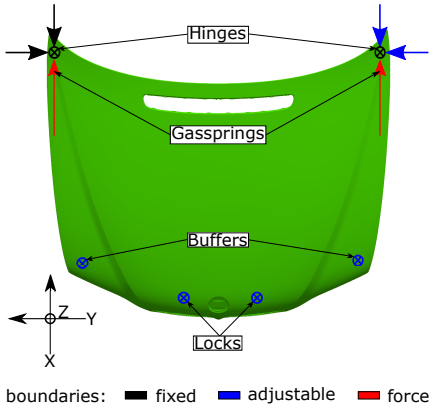


Figure 3 Sheet metal car body part used in automotive industry. The used boundaries for the simulation are high-lighted, with external forces used via gas springs. The coordinate system for this part is shown.

For a more application-driven view, a real-world example from the automotive industry was chosen. The geometry is an engine hood. This part has two hinges, two locks and two buffers as mechanical boundaries attaching the hood to the chassis, see Figure 3. A finite element (FE) simulation predicts deflections during the assembly process of the part.

However, the final shape of a material part can vary due to production uncertainties. To deal with uncertainties and tolerances during the production, the engine hood's boundaries, i.e., hinges, lock and buffers, are adjustable. Adjusting these boundaries properly to obtain an acceptable gap and flushness is a challenging task [13]. A post-assembly measurement induces necessary corrections. The goal of the proposed method is to find the best set of changes from measured deviations, which forms the optimal set of adjustments. The method uses as input an assemble of statistical distributed simulations that cover the solution space spanned by the available adjustment possibility of each boundary. The used car hood is an assembly containing seven individual sheet metal parts, connected by spot welds and different types of adhesives. Based on the CAD files, a simulation model was created by meshing the geometry with 3D-shell elements and connecting the assembly considering spot weld, adhesive positions, and thicknesses of components. The material model is linear-elastic with an e-modulus of 210Gpa and a Poisson rate of 0.3. Two fixed, external loads, modeling the gas springs near the hinges with the magnitude of 580N each, complete the model.

3.2 Image Database Linearization

We linearize the depth image database by enumerating the Cartesian product parameter space using an arbitrary ordering of axes. We hypothesize (and empirically confirm, cf. Section 5) that ordering axes in a manner that the fastest axis (along which subsequent images lie in the linearization) should be chosen such that the inter-image similarity is large to benefit compression efficiency.

Most video codecs assume that the changes from one frame to the other happens to be smoothly and in a predictable way over a longer period (see the preset parameters [10]). Therefor we assume that the best compression performance results will occur for a smooth ordering of the images. We are investigating the impact of camera path, data time step, ensemble parameter and visualization parameter as different approaches for the dominating ordering axis (see Fig.4).

Camera Location

The camera parameters are usually the first parameters that are used while creating an image data base. A collection of images ordered by camera angles is similar to tracking shots in video sequences, which are the target of video compression techniques. Based on the number of viewpoints that are needed to capture a visualized data set, however, it is not necessarily the axis with the smoothest transition between successive images. Especially if storing a minimum amount of images needed to visually adequately reconstruct geometry [20], camera location index as the fastest axis is not necessarily a good choice.

Data Time

For unsteady data sets, time provides a natural parametrization. In practice, memory or I/O constraints determine the step size between successive discrete time points. Furthermore, considering ensemble data, step sizes may vary per ensemble member. The chosen visualization techniques also plays a crucial role here, while some techniques will result on bigger visual changes with small changes in the time, others may behave in the opposite direction. For our setup, we therefore do not rely on smooth transitions between images when choosing the time as the dominant axis.

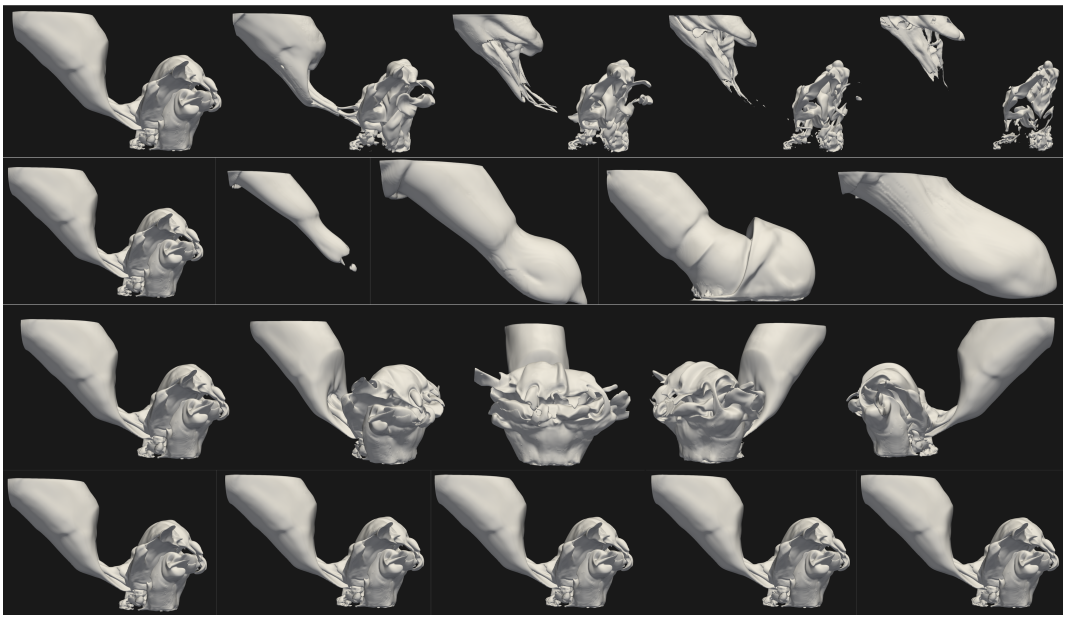


Figure 4 Showing different ordering strategies for one data set. Depending on which axis is chosen to be the dominant one, the similarity between successive images could vary dramatically. From top to bottom: Iso-value, Ensemble Parameter, Camera rotation, Simulation time step.

Ensemble parameter space

In ensemble data sets we are particularly interested in the differences that relate to changes in the input parameter space, which leads to similar effects as for setting the time step size mentioned above. For our investigation, we therefore expect that it behaves similar to the time ordering for unsteady data sets if they have a matching resolution of the step sizes.

Isovalue

Concerning algorithmic parameters that change the visualization result, we consider isovalue in the cases described above as a proxy for more general settings. Most notably, it is one of the most often changed parameters when exploring a data set. In general, variation between images should be small under a fine-grained sampling, as the surface varies slowly and smoothly with change in isovalue. Thus, we hypothesize that this parameter is a good choice for the fastest axis in the linearization if the sampling is not too coarse.

3.3 Depth Image Encoding

Before applying video compression to depth images, these must be transferred into a format that is suitable for ingestion by video codecs. While depth images encode a single scalar in the range $[0, 1]$ per pixel, the lack of a robust support for single-channel image formats in video encoders (e.g. the gray-scale format *12greyle*) makes it difficult to pass the depth values directly. Furthermore, a lack of format with high-bit depth – at most 12 bits – would induce inacceptable quantization to the depth images and make them essentially unusable for compositing.

Hence, we encode the scalar field (depth image) into an image format with 3 channels and 8 bits per channel using Morton coding [22], mapping a 24-bit depth value into 3x8 bits. Here, Morton coding is much preferable to simple mapping of the high, middle, and low bytes

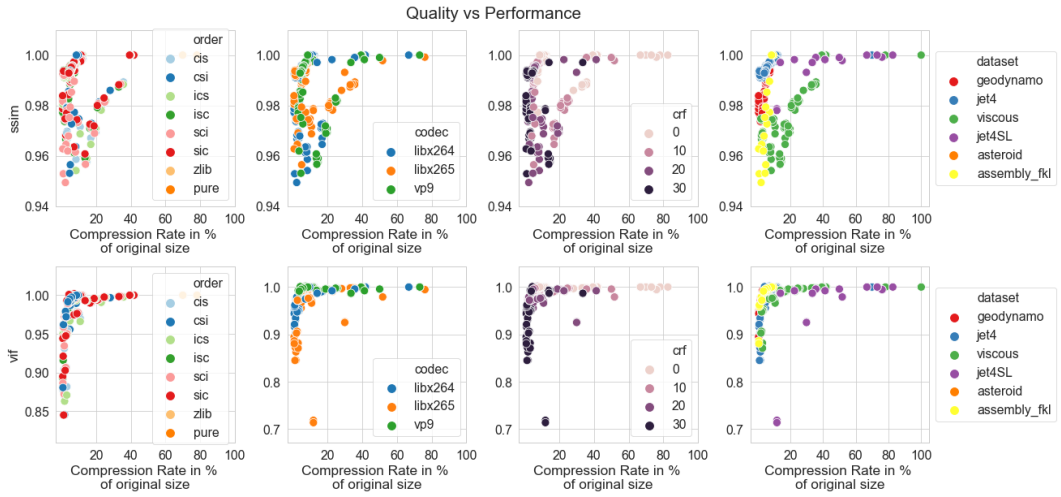


Figure 5 Comparing the results for the compression performance with the image quality. Top row: SSIM. Bottom row:VIF[24]. F.l.t.r. clustered by image ordering (s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering streamlines, used video codec, constant rate factor (crf) and data set. The sweet spot is located in the top left corner, where image quality and compression rate delivers the best performance.

to three channels, since the Z-order curve underlying this coding guarantees that close-by depth values will be mapped to close-by tuples. The effect of this mapping, interpreting the three channels as red, green, and blue colors, is illustrated in Figure 6. We pass the result of this mapping to the video codec in the *yuv444p* format, as YUV is the natural color space in which all considered codecs operate and errors induced through in-codec color space conversion can be avoided. Note that video codecs typically accept various formats that represent color information at reduced resolution when compared to luminance information, such as e.g. the often used *yuv420p* format. However, we do not see an indication at this time how such sub-sampling would benefit the compression of depth images and thus only consider *yuv444p*. In general this process is applicable to any scalar field (not only depth) input on the images.

3.4 Video Compression

Following linearization and encoding, we pass the *yuv444p* image sequence directly to the video codec. As a general interface to different codecs, we employ the `ffmpeg` [14] tool, and encode using three major and broadly available video codecs:

- H.264 [26] The H.264 is using motion estimation to minimize temporal and spatial redundancies. It is classified as a block-oriented motion compensating compression technique.
- H.265 [11] The H.265 is based on the same principle than the H.264. Its main difference is the increased coding tree unit from 16×16 to 64×64, though leading in general to higher compression rates.
- VP9 [15] The VP9 codec is also a block-based format. Its main application area is for web streaming and therefor it is designed to ensure a certain bit-rate rather than a constant quality like H.264 and H.265.

Codec Parameters

In general there is a very large number of parameters that affect each encoders and allow tweaking it to different types of input. To keep our study reasonable and achieve comparable results, we opt to focus on the *constant rate factor* (**CRF**) as the central parameter that effects the amount compression for each codec, and we consider the values 0 (lossless), 10, 20, and 30 to represent different levels of compression. Typical CRF choices for natural images are in the range from 18 to 24.

All codecs furthermore support variable-rate encoding which adapts bandwidth used based on heuristics, in the interest of overall better image quality. However, as the assumptions underlying these heuristics are geared towards natural images and are not well documented, we choose not to examine this mode due to the large amount of unpredictability it induces on results. Note that the VP9 codec is primarily intended for streaming applications, and thus constant rate encoding is not its optimal mode of operation; however, we still consider it in this study to to its ubiquitous use an generally good compression / quality performance.

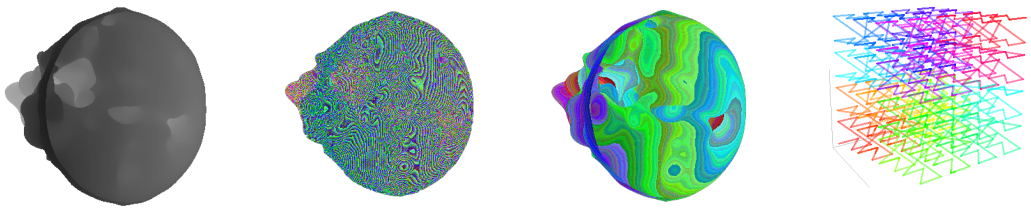
The H.264 and H.265 codecs are further able to trade off compression speed against image quality through a preset choice. We here use the *veryfast* preset to achieve the best speed. While image quality would improve with slower presets, at the expense of much longer compression times, we obtain a lower bound on image quality, which we consider to reflect real-world considerations best. A summary of the parameters steered by these presets can be found in [10]

As the output of the compression process, we obtain a single video file that represents the entire visualization depth image database.

3.5 Image Retrieval

To retrieve images, we again employ the `ffmpeg` tool to retrieve a single image from the video file in *yuv444p* format, and decode the Morton ordering.

Retrieved images are compared against the (uncompressed) ground truth images, using the metrics described in the following section.



(a) Ground Truth. **(b)** Modulo. **(c)** Morton. **(d)** Z-order curve.

Figure 6 Encoding of a 24-bit depth image (a) to 3x8 bit channels using reinterpretation as three bytes (b) or Morton coding (c). The three resulting channels are interpreted as RGB for illustration purposes. (d) shows the Z-order curve underlying the Morton coding.

4 Evaluation Metrics

For our evaluation we distinguish three major factors: Quality of the reconstructed visualizations images, compression efficiency in the sense of file size and time to retrieve an image in the original domain, and, especially for surfaces, how much the errors introduced by compression affect the compositing of two or more visualizations post hoc.

4.1 Image Quality Metrics

For our quality benchmark we choose a set of numerical and image quality measures. Typically the raw numerical metrics such as e.g. mean squared error are hard to interpret. We therefore consider quality measures for images as well. Table 1 provides an overview of all employed quality metrics.

Table 1 Used Image metrics for the quality benchmarks.

Metric	Description	Value Range (identical value)
MSE	mean squared error	$\geq 0(0)$
PSNR	peak signal to noise ratio	$\geq 0(\infty)$
MAE	mean absolute error	$\geq 0(0)$
SSIM	structural similarity index	$[0, 1](1)$
VIF	visual information fidelity measurement [24]	$[0, 1](1)$

4.2 Performance Metrics

For the performance measure we use the compression and retrieval rate. For the compression rate we use the highest z-lib compressed depth images as our base value. All values are then relative increases or decreases in percent. For the retrieval rate we randomly draw 300 frames from each video and apply the morton inversion to them to end up with depth images. We measure the time from the start of the retrieval (video already loaded) to the end of the conversion. For comparison we measure the time for 500 database calls using Paraview and the ttkCinema filter. The measures are then normalized to time to retrieve one image. We do not take speed ups from parallel computing into account, as such practical values can differ.

4.3 Compositing Benchmark

In the compositing benchmark we test how robust the video compression is for small changes in the depth value, that can lead to cluttering effects when compositing multiple surfaces in a post-processing step. This test is only done for a set of previous selected surface pairs from the Jet flow data set, representing a worst-case scenario from practice. Error is measured in the number of wrongly assigned pixel of the different surfaces to fore- and background. We do the compositing once for the ground truth depth images and once with the compressed images and then compare the two results.

5 Results

We have investigated a total of 264 encoded videos from 5 different data sets and three visualization types. In general, the video compression always achieves higher compression rates than *zlib*(see Fig.7), which we use as a general purpose compression technique and practically relevant comparison baseline. In comparison to `zlib` however, retrieval times were increased.

The H.264 codec turns out to be the most predictable, exhibiting the fastest encoding speed and generally low error. While VP9 resulted in a less overall error the encoding time is about 3 times higher for lower rate factors(see Fig.7 bottom). Regarding the constant rate

factor, as the dominant parameter for all codecs, we found out that a value around $crf = 20$ is the best trade off between compression rate and error (see Fig.5, column 4). The sweet spot is best defined as the upper left corner in Fig.5. For this parameter combinations we observe the best quality with the highest amount of compression.

Considering resulting images (Fig.1), we can observe different visual qualities for different application scenarios. With increasing CRF value the video compressor starts deleting pixels in areas with similar values. For the streamlines these areas are much smaller, which explains why the compressed file size for these is much higher.

For the surface compositing test, no significant errors can be reported. We counted the number of wrongly assigned pixels in the compositing steps. For crf = 0, no wrongly assignments were counted. For crf = [10,20,30] the maximum were 18 wrongly assigned pixels. In relation to the image size of 512 x 512 this results in a relative maximum error of 6.9×10^{-6}.

5.1 Image Quality

We can observe a predictable increase in the error rates with increasing constant rate factor. Over all settings the mean SSIM is above 0.98. Referring to Baker et.al. [5] this is still in the range were most people wont see a difference. It can be seen in Fig.1 that the outliers are nonetheless visible, but a simple smoothing may overcome this issues, as the outliers are mostly single isolated pixels. There is no significant influence of the image ordering on the resulting error rates for both iso-surface and streamline visualizations(Fig.10, row 2,3). But for the assembly use-case we can observe an effect of the ordering in the PSNR and SSIM measures. Here the quality for ordering first along the simulation index and than the camera view port (CS) achieves better results. In the assembly use-case we observe a stronger decline in image quality with increasing constant rate factor than in the other examples (see 6a). The H.265 codec is the only one which mean error is not zero for the lossless setup, resulting in the fact that it is not able to achieve a complete lossless compression (Fig.10, row 1). For higher crf values the difference between the codecs gets smaller. Regarding the image metrics the H.264 codec is overall the best performing followed by the H.265 for crf 10 and 20. For crf 30 all codecs are nearly equal. But overall the VP9 codec results in less noisy images, as it always outperforms the other codecs in the peak signal to noise ratio ($psnr$). In Fig.10 row 4, we can see that the original data set has a strong influence on the achievable image quality. Additionally the image quality for streamline visualization is more sensible to an increased rate factor than the (iso-)surface visualization (see Fig.1, row 4, blue and purple). Applying the visual information fidelity measure (vif) results in different ratings especially for the codecs and data sets (Fig.10,column 4, row 2 and 4). The mean absolute error is nearly equal among all settings and therefor no suitable measure.

5.2 Compression Rate

The compression rate of the lossless video encoding is in the order of 5 times smaller than the original uncompressed data and about 3 times smaller than zlib compression with the highest settings (Fig.7, row 1 and 2, columns 1). For lossy compression we achieve up to 40 times smaller sizes than the original data (Fig.7, row 1 and 2, column 4). The H.264 and H.265 codec achieved the highest compression results with increasing difference to VP9 with increasing rate factor (up to 3 times for $crf = 30$,see Fig.7, row 1 and 2). For the lossless compression the H.265 codec is the best performing. But we have seen in the quality benchmarks, that this codec is not able to achieve real lossless compression which leads to

an unfair advantage in this comparison. For crf 10 the H.264 codec is about 30% better than H.265 and VP which are fairly on the same level. For crf 20 the difference between H.264 and H.265 is getting smaller while VP9 does only small progress. Overall the H.264 and H.265 have a quadratic negative slope for the compression rate based on the crf value, while VP9 has a near linear slope. The first axis of image ordering has a small impact on the compression rates in the order of $2-4\%$ in absolute scale(rel. to each other $18-24\%$). The effect of the second axis is negligible. A big difference in compression rates between isosurface and streamline visualization can be observed.For the lossless setting the compression rates are near to the zlib compression. Compared to the iso-surface visualizations, the compression rate have a less steeper slope. For $crf = 30$ the compression rates are still up to 5 times higher than for iso-surfaces. The difference in the codecs also persists in this setup.

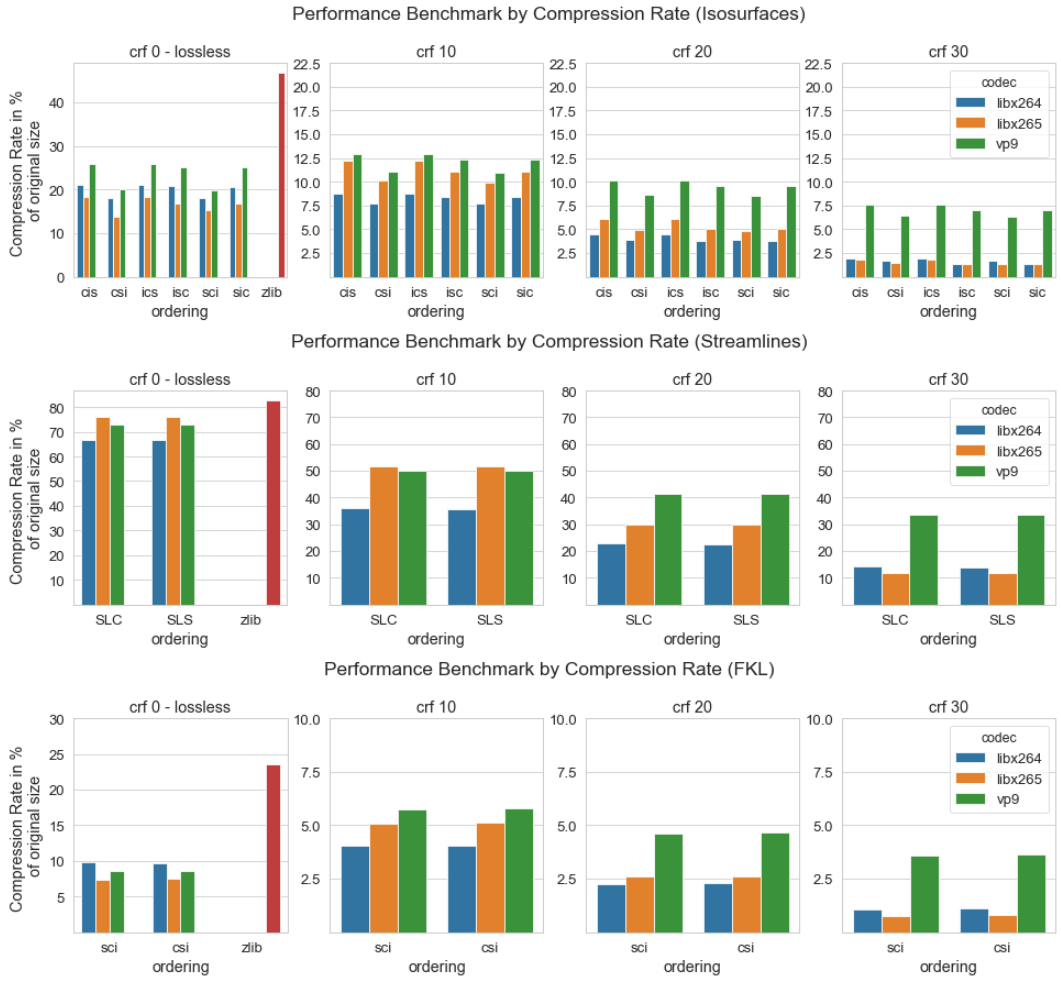


Figure 7 Results for the compression performance. Top rows: Compression rate in % of the original uncompressed image data base. grouped by image ordering for iso-surfaces(s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering for streamlines (S: camera **s**phere, C:**c**ycle time step) and constant rate factors, colored by the used codec. Bottom row: Encoding time per image.

5.3 Encoding and Decoding

The encoding was performed on one node of a cluster with a Intel® Xeon® Processor E5-2640 v3. The encoding time per image was in the range of 20 ms for loss-less compression to

nearly 400 ms for the highest chosen compression rate. The rendering of the images for the database was performed on multiple machines to achieve a suitable large amount of images for the presented study. For in-situ we assume, that the data is already loaded in memory. As such only the time to extract the iso-surface and render the depth image is of relevance. In a small benchmark for images with a size of 512×512 we end up with a mean rendering time per image in the order of 72 to 230ms and a pipeline execution time for the iso-surface extraction in the order of 310 to 620ms, depending on the visualized structures complexity. The remaining driver thus is the I/O. Todays storage bandwidth is in the order of 6 -12 GB/s. The image sizes in our resulting data base are in the range of 260 KB to 295 KB. This results in 0.022 to 0.049ms per image and thus is neglectable. In a rough comparison the encoding time for the video compression adds a factor of 0.05 to 0.47 to the computation time in relation to the already performed rendering pipeline. The encoding time per image

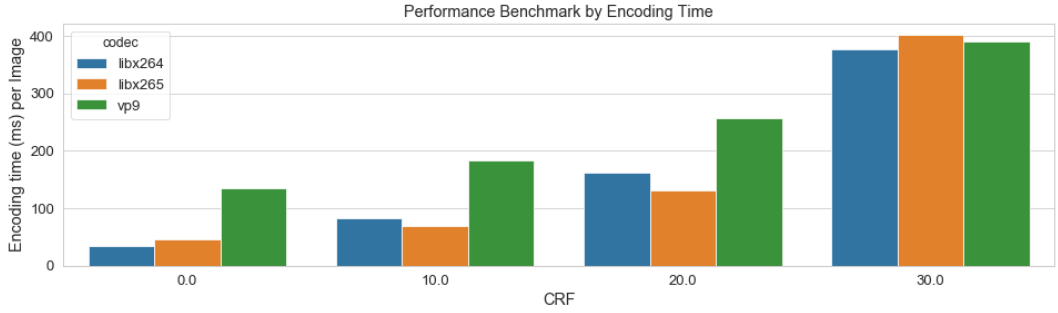


Figure 8 Results for the Encoding time per image, grouped by codec. With increasing constant rate factor (CRF) the differences between encoding times is equalizing.

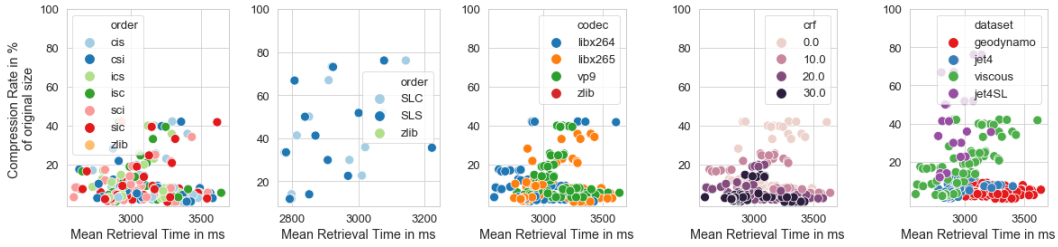


Figure 9 Comparison of compression rate and retrieval time clustered by codec, image ordering for iso-surfaces(s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering for streamlines (S: camera **s**phere, C:**c**ycle time step), constant rate factor and data set.

has a nearly quadratic slope based on the crf value (Fig.7, row 3). For the codecs we can see that VP9 takes about 3 to 4 times longer, for low crf values, than H.264 and H.265. With increasing crf values the differences in encoding time between the codecs is decreasing. For lower crf values, the H.264 and H.265 codec outperform the VP9 codec. As there exist a huge numbers of hardly controllable side effects when measuring the execution time, we can make only general observations. Based on the streamline image orderings *SLC* and *SLS* compared to the orderings for the iso-surface images (*cis,csi,ics,isc,sci,sic*), we can see that the resulting compressed file size and the time to encode are negatively correlating (cor= -0.342, Pearson).

The decoding and retrieval was performed on a local desktop machine, to avoid bandwidth and network effects in the measurement. For the mean decoding or retrieval times based on random access we end up in the interval of 2.6 to 3.8 seconds. No effects for the ordering and crf value on the retrieval time can be observed (see Fig.9). There exist a slight negative correlation (cor= -0.39789, Pearson) between the size of the compressed video and the retrieval time, which means that higher compressed videos have higher retrieval times. Note that this correlation does not directly cope with the chosen crf value. For the codecs we can find some clusters which are leading to a weak order based on the retrieval times (H.264<H.265<VP9). Further Fig.9 row 5 suggests that the underlying data sets form distinguishable clusters. For the quality benchmarks, we have encountered the same effects. We found out that our measured retrieval times are about 4 to 5 times higher than using the implementations from OpenCV for frame retrieval. This implies that a faster implementation for image retrieval is possible in principle. Nonetheless the implementation from OpenCV is targeted towards the retrieval based on time stamps and not exact frame IDs. A such the retrieval is not exact, but rather chooses the next frame in a predefined range near the target.

5.4 Overall Observations

Based on five data sets with very different properties we identified that the H.264 codec delivers the overall best performance regarding image quality, compression rate and retrieval time. For encoding time it is on the same level as the slightly faster H.265 codec. Another advantage is that the H.264 codec is able to achieve true lossless compression. Regarding the investigated parameters we found that only the fastest axis chosen for linearization is significant, and the ordering of the remaining axes does not impact the results. The recent study from Baker et.al. [5] provides a guideline for choosing a good threshold on typical image metrics. To this point the most predictable one is the structural similarity metric (SSIM). For the climate model they investigated a value of 0.99995 is needed for identical visual quality, which can be achieved by choosing the constant rate factor to be 0. We determined that a value around CRF=20 delivers the best trade off between compression rate and visual quality, regarding a threshold on SSIM of 0.98 determined on medical literature by [3]. In other words, for a successful application of video compression for visualization image data bases, the only parameters a user has to determine is the fastest linearization axis along which images share the most similarity, and which quality the resulting images should satisfy. Applying video compression is not restricted to only surface-based visualization but also works for line-based visualizations, which is shown with the jet flow streamline example, but comes with decreased compression performance. In general, unsurprisingly, we observe that the compression works best for less cluttered visualizations with smooth areas and few edges.

6 Conclusions

We have provided a study concerning the applicability of lossy video compression to visualization image databases. Our findings confirm observations made by Berres et al. [7], thereby strengthening further the argument that video compression is a viable and beneficial approach leading to very good compression rates when compared to a general-purpose lossless compressor. Further, the lossy compression approach introduces only a minor image quality deterioration, and, in some case, does not lead to loss at all. We have shown that the complexity of an implementation is manageable. Furthermore, the reduced file sizes generated via video compression make generally possible the exploration of larger parameter

spaces, greatly benefiting a computational scientist when analyzing a visualization image database. Equipped with suitable meta-data, we believe that video file formats could serve as effective containers for visualization image databases generally, thus simplifying database management tasks.

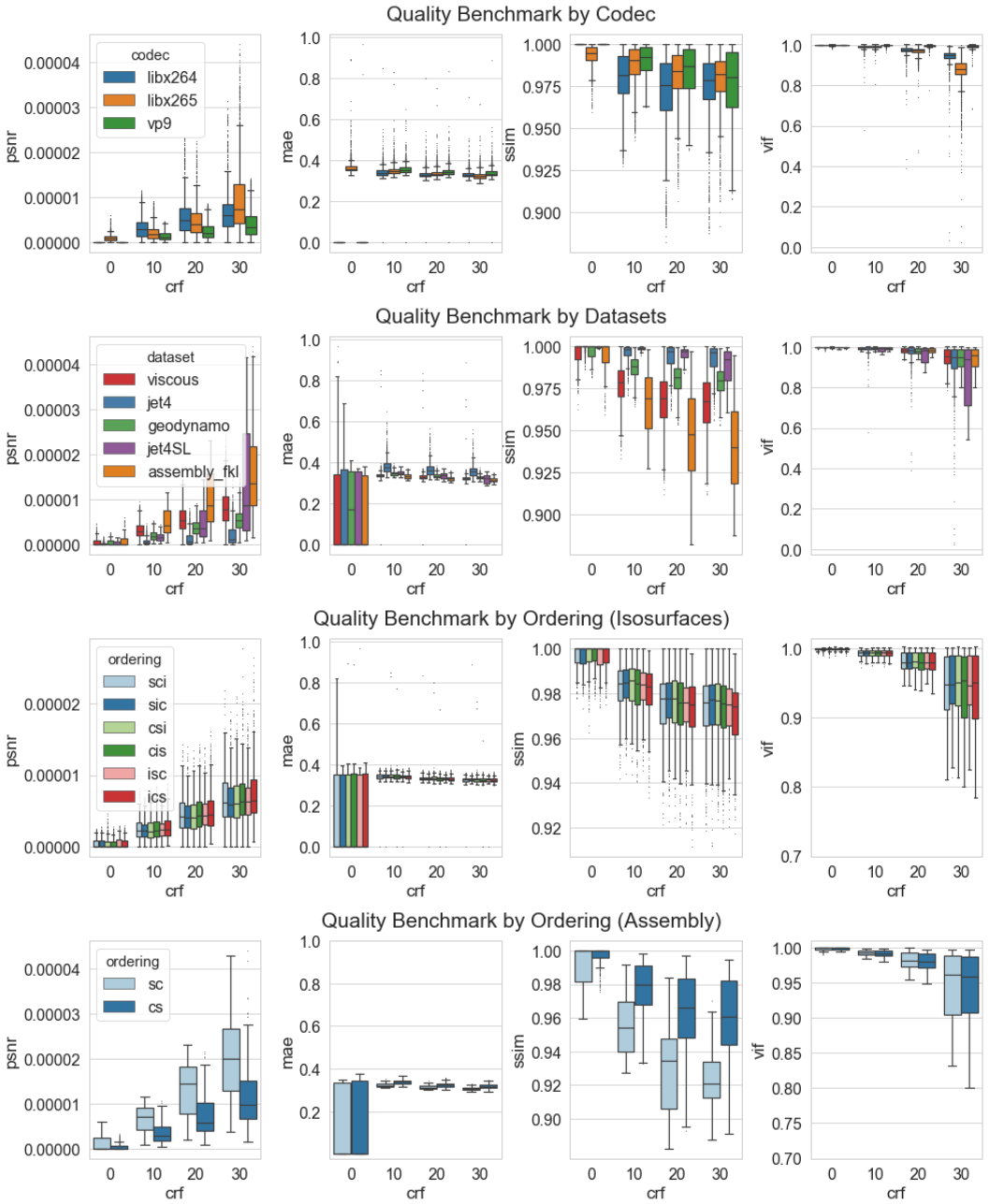


Figure 10 Results for the quality benchmark. Box-plots from top to bottom grouped by codec, image ordering for iso-surfaces(s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering for streamlines (S: camera **s**phere, C:**c**ycle time step) and data set.

In general, the ordering only effects the compression with primary order parameter. Thus it is sufficient in most cases to determine the ordering axis with the highest expected similarity between subsequent images. For the image quality, we only observed an effect for the assembly

use case and there only for the SSIM and PSNR. In this setting we achieved in general higher compression rates due to the very high similarity between the images. Thus we assume that the effect of ordering on the image quality increases with higher compression rates. The use of lossy general-purpose compression techniques – i.e., techniques not primarily aimed at video compresssion – such as the ZFP [19] compressor, should be investigated. Additionally, an important aspect for many real-world applications is the ability to perform video compression and encoding in parallel for in situ visualization purposes. We believe that independent encoding of subsequences is feasible. Moreover, utilizing hardware-enabled acceleration made possible, for example, by encoding via GPUs could further improve compression performance substantially. It would also be of interest to determine a near-perfect order used to generate all in situ visualization mages of a scientific data set. If one were able to optimally define the trajectory of the "virtual camera" used to produce in situ visualization images, then standard image compression codecs would be highly effective, due to the high degree of frame-to-frame coherence for such a camera trajectory. It is planned to consider these aspects in future research.

References

1 James Ahrens, Sébastien Jourdain, Patrick O'Leary, John Patchett, David H. Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 424–434, Piscataway, NJ, USA, 2014. IEEE Press. doi:10.1109/SC.2014.40.

2 Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, pages 25–29. ACM, 2015.

3 A. H. Baker, D. M. Hammerling, S. A. Mickelson, H. Xu, M. B. Stolpe, P. Naveau, B. Sanderson, I. Ebert-Uphoff, S. Samarasinghe, F. De Simone, F. Carbone, C. N. Gencarelli, J. M. Dennis, J. E. Kay, and P. Lindstrom. Evaluating lossy data compression on climate simulation data within a large ensemble. *Geoscientific Model Development*, 9(12):4381–4403, 2016. doi:10.5194/gmd-9-4381-2016.

4 Allison H Baker, Dorit M. Hammerling, and Terece L. Turton. Evaluating Image Quality Measures to Assess the Impact of Lossy Data Compression Applied to Climate Simulation Data. *Computer Graphics Forum*, 2019.

5 Allison H. Baker, Dorit M. Hammerling, and Terece L. Turton. Evaluating Image Quality Measures to Assess the Impact of Lossy Data Compression Applied to Climate Simulation Data. *Computer Graphics Forum*, 2019.

6 Janine C. Bennett, Hank Childs, Christoph Garth, and Bernd Hentschel. In Situ Visualization for Computational Science (Dagstuhl Seminar 18271). *Dagstuhl Reports*, 8(7):1–43, 2019. doi:10.4230/DagRep.8.7.1.

7 Anne S. Berres, Terece L. Turton, Mark Petersen, David H. Rogers, and James P. Ahrens. Video Compression for Ocean Simulation Image Databases, 2017. doi:10.2312/envirvis.20171104.

8 Tim Biedert and Christoph Garth. Contour tree depth images for large data visualization. In *Eurographics Symposium on Parallel Graphics and Visualization, Cagliari, Sardinia, Italy, May 25 - 26, 2015.*, pages 77–86, 2015. doi:10.2312/pgv.20151158.

9 Tim Biedert, Peter Messmer, Thomas Fogal, and Christoph Garth. Hardware-accelerated multi-tile streaming for realtime remote visualization. In *EGPGV 2018: Eurographics Symposium on Parallel Graphics and Visualization, Brno, Czech Republic, June 4, 2018*, pages 33–43, 2018. doi:10.2312/pgv.20181093.

10 Joey Blake. FFmpeg preset files. Contribute to joeyblake/FFmpeg-Presets development by creating an account on GitHub, May 2019. original-date: 2010-09-12T13:01:27Z. URL: `https://github.com/joeyblake/FFmpeg-Presets`.

11 Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, Chi Ching Chi, Tobias Mayer, Ben H. H. Juurlink, and Thomas Schierl. Hevc real-time decoding. In *Optics & Photonics - Optical Engineering + Applications*, 2013.

12 David Ellsworth, Bryan Green, Chris Henze, Patrick Moran, and Timothy Sandstrom. Concurrent visualization in a production supercomputing environment. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):997–1004, 2006.

13 H. Hagen F. Claus, F.-A. Rupprecht. Online simulation considering production uncertainties to improve assembly quality. In *NAFEMS World Congress*. NAFEMS, 2019. URL: `https://www.nafems.org/publications/resource_center/nwc_19_356/`.

14 FFmpeg Developers. Ffmpeg. URL: `https://ffmpeg.org/`.

15 Adrian Grange, Peter de Rivaz, and Jonathan Hunt. VP9 Bitstream & Decoding Process Specification v0.6, 2016.

16 M. Hummel, R. Bujack, K. I. Joy, and C. Garth. Error estimates for lagrangian flow field representations. In *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*, EuroVis '16, pages 7–11, Goslar Germany, Germany, 2016. Eurographics Association. `doi:10.2312/eurovisshort.20161153`.

17 Akira Kageyama and Tomoki Yamada. An approach to exascale visualization: Interactive viewing of in-situ visualization. *Computer Physics Communications*, 185(1):79–85, 2014.

18 S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum*, 37(6):422–447, 2018. `doi:10.1111/cgf.13336`.

19 Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014.

20 Jonas Lukasczyk, Eric Kinner, James Ahrens, Heike Leitte, and Christoph Garth. A View-Approximation Oriented Image Database Generation Approach. *IEEE 8th Symposium on Large Data Analysis and Visualization(LDAV), 2018*, page 11, 2018.

21 K. Ma. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, November 2009. `doi:10.1109/MCG.2009.120`.

22 Guy M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company, 1966.

23 Patrick O'Leary, James Ahrens, Sébastien Jourdain, Scott Wittenburg, David H. Rogers, and Mark Petersen. Cinema image-based in situ analysis and visualization of mpas-ocean simulations. *Parallel Computing*, 55:43–48, 2016. Visualization and Data Analytics for Scientific Discovery. `doi:10.1016/j.parco.2015.10.005`.

24 Hamid R. Sheikh and Alan C. Bovik. Image information and visual quality. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–709. IEEE, 2004.

25 Bong-Soo Sohn, Chandrajit Bajaj, and Vinay Siddavanahalli. Volumetric video compression for interactive playback. *Computer Vision and Image Understanding*, 96(3):435–452, 2004.

26 Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra. The h.264/avc advanced video coding standard: overview and introduction to the fidelity range extensions. In *SPIE Optics + Photonics*, 2004.

27 Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE transactions on visualization and computer graphics*, 24(1):832–842, 2018.

28 Anna Tikhonova, Carlos D. Correa, and Kwan-Liu Ma. Explorable images for visualizing volume data. In *IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan,March 2-5, 2010*, pages 177–184, 2010. `doi:10.1109/PACIFICVIS.2010.5429595`.

29 Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

30 J. Woodring, S. Mniszewski, C. Brislawn, D. DeMarle, and J. Ahrens. Revisiting wavelet compression for large-scale climate data using jpeg 2000 and ensuring data precision. In *2011 IEEE Symposium on Large Data Analysis and Visualization*, pages 31–38, October 2011. doi:10.1109/LDAV.2011.6092314.