Automatic Camera Selection for In Situ Visualization

Nicole Marsaglia
Area Exam

Abstract—**We consider the problem of automatic camera selection in the context of in situ visualization. This problem is important because high-performance computing trends are increasingly mandating in situ processing, and this processing paradigm frequently has no human-in-the-loop — new research is needed to automate the decisions that have previously been made by human beings. We begin by briefly evaluating what makes an image good, i.e., informative, pleasing, etc. The majority of this work is in surveying existing techniques for camera selection that have been considered in a non-in situ setting, organizing them around geometric- and data-driven techniques. We then survey considering the in situ context that an automatic camera selection algorithm should run, specifically in the infrastructures that house such algorithms and in the driving use cases from application codes. Finally, we conclude the survey by considering data sets from representative simulation codes and evaluating the efficacy of various existing camera selection techniques.**

1. Introduction

As the computational power of computers steadily increases, researchers' abilities to run more complex simulations at higher resolutions also increases. And the growing gap between compute power and I/O has made it very difficult for scientists to interact with their data. In the traditional paradigm, scientists would save out their data at regular or irregular intervals and explore the results post-hoc. To combat the growing I/O gap, scientists would simply save out their data less and less frequently, but this runs the risk of missing important phenomena, or discovering the phenomena but lacking the temporal resolution to truly explore it. This has resulted in a new in situ processing paradigm. In situ processing means that data is not written to disk, instead data is processed in place, in many cases sharing resources with the simulation. There are several strategies researchers can employ with in situ processing, and it depends on if they have *a priori* knowledge of their data. If there is a priori knowledge, sceintists can visualize their data as the simulation is running, but again the scientist runs the risk of missing important phenomena. If there is no a priori knowledge, scientists can perform lightweight analysis to process data as the simulation is running, allowing scientists the ability make decisions based on the analysis or save a reduced/transformed version of the data to disk for post-hoc exploration. Alternatively, instead of saving transformed/reduced data for post-hoc exploration, scientists can perform automatic scientific visualization. That is, perform in situ processing to automatically analyze, reduce, or transform the data without a human-in-the-loop driving the exploration.

And yet, with supercomputers on the brink of exascale capabilities, new in situ analysis and visualization methods are still needed. Ideally they need to be lightweight, not increasing the simulation runtime too dramatically or requiring too much from the memory footprint. An emerging requirement is that the new algorithms should run independently. That is, algorithms that used to require user input should be able to generate input automatically, thus eliminating the human-in-the-loop. Research in automatic visualization is being developed to solve a number of problems, such as automatic seed placement for flow visualization [LS07], [MCHM10] and automatic transfer functions for volume rendering [WDC*07], [RBB*11], [VFSG06], [VKG05], to name a few. The primary focus of this paper is automatic camera placement while running in situ. The primary focus of this paper is automatic camera placement.

Visualization is a key component to understanding large scale scientific data. Compared to raw data, images are perceivable to the human mind, allowing us to visually discover phenomena, detect patterns and trends, as well as outliers and possible errors. Additionally, images can be used as proof of concept by clearly conveying the conducted research or results. But not all images are created equal: data can produce countless images depending on the camera placement, and while some may contain valuable information, others may not. Being able to determine the best viewpoint based on some criteria is a useful way to produce noteworthy visualizations. But with the increase in computing power, scientists are having to adapt how they visualize their data and how to produce the best representative image of their data. And with the increasing size of simulation data, where to point the camera without a human-in-the-loop is an ongoing problem.

In the post-hoc setting, algorithms for determining the best viewpoint has been a growing area of research and has been used in scene exploration and camera placement, image-based modeling and rendering, scientific visualization, shape retrieval, and mesh simplification [BFS*18]. The majority of the techniques perform some calculation based on the geometry of the data, with few techniques for field data.

This paper explores the automatic camera placement techniques that have been proposed to date, as well as presenting preliminary evaluations of more than half of the metrics applied to scientific data in order to test their fitness for in situ analysis.

The paper is organized as follows: Section 2 explores the question, "What makes an image good?"; Section 3 describes the metrics used to quantify the quality of a viewpoint; Section 4 describes software packages and libraries that enable in situ analysis and visualization; Section 5 explores in situ analysis and visualization use cases on large-scale simulations; and Section 6 evaluates a number of the viewpoint metrics.

2. What Makes an Image Good?

Trying to quantify what makes an image good is not a new venture. The Ancient Romans did it with the *Golden Ratio*, a proportion that is still considered to be visually pleasing, and psychology has shown that it is preferred when formatting an image [GRMS01], [Arn88], [SK32].

In the 1930s a mathematician named George D. Birkhoff attempted to quantitatively measure beauty [Bir33], [Bir56]. He believed beauty is a ratio of order over complexity, i.e. beauty increases as complexity decreases. But after applying these notions to a wide array of formats, including simple geometries, poetry, melodies, art, etc., he was unable to produce a general formula for order and complexity. And while he did not produce a detailed equation for beauty, he did remark that "a fine composition is always arranged so as to be easily comprehensible."

Tarr and Kriegman [TK01] conducted psychoanalysis experiments investigating the influence of an object's aspect on user preference. They found that for many models there exist a small number of views that are preferred by most people.

It has also been shown through numerous user studies that users prefer an image with a three-quarter or canonical view [BTB99], [PRC81]. Kamada et al. [KK88] calls that a non-degenerate view and consider an image good if it minimizes the number of degenerate views, as shown in Figure 1. According to Blanz et al. [BTB99], canonical views are stable and expose as many salient and important features as possible. But while these views have been proven to be visually pleasing, they provide no guarantee of scientific merit or importance.

Other aspects of an image are also critical to user understanding and preference. For instance, color maps play a key role in image comprehension. Bujack et al. [BTS*18] survey research that quantifies good color maps and present mathematical design rules for choosing color maps.

There has also been work on optimal light source placement. Much like color this is a non-trivial problem that may not have a general solution [JPP02], [PF92], [PRJ97]. Current works are based on inverse lighting techniques, where the optimal light source is deducted from an expected result. In general, most current techniques are not fully automatic and require user interaction [MAB*97], [HM03]. Gumhold [Gum02] presents an automatic method based on *light entropy*, but results are not consistent.

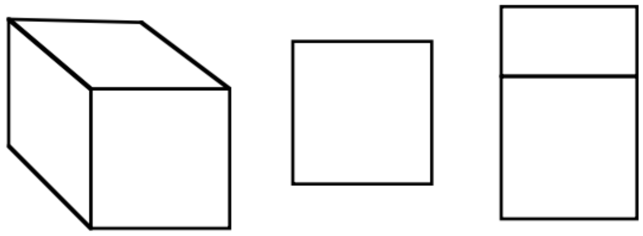


Figure 1: Taken from Barral et al. [BDP99], this image depicts one non-degenerate view of a cube (left) and two degenerate views of a cube (center and right) as defined by Kamada et al. [KK88]. Notice that the non-degenerate cube is shown at a three-quarter, or canonical view.

Bordoloi and Shen [BS05] understand that the "best image" depends on context. From a volume rendering perspective, they have two guidelines for determining a "good image":

- A viewpoint is good if voxels with high noteworthiness factors have high visibilities. This guideline applies to user input that has placed importance on certain aspects of the model when defining the transfer function.
- A viewpoint is good if the projection of the volumetric data set contains a high amount of information.

It is clear that there is a significant amount of research into what makes an image good, but this raises the question: "How do we find it?"

2.1. How to Find the Best Image?

Finding the best image of a data set is non-trivial problem. For instance, a data set could contain multiple "best images" depending on what the researcher wants to convey. Polonsky et al. [PPB*05] treats viewpoint selection as a function where the best viewpoint maximizes this function. A function that measures the viewpoint is called a *view descriptor*. Polonsky et al. based their view descriptors off of the following three principles:

- **Geometric Complexity.** The first principle is based on the geometry of the scene and will assign higher scores to views that expose as much of the geometric complexity as possible.
- **View-dependent Features.** The second principle focuses on features that are view-dependent. In this case, there are features that are only visible from certain views; these views are considered better viewpoints.
- **Primitive Elements** The third principle involves the elements that are assigned values or otherwise used within the descriptor. Descriptors can use a number of primitive elements (vertices, faces, edges etc.) to determine the best viewpoint, this principle also considers larger portions of the model that have some significance or meaning.

Depending on the data set or what the researcher wants to convey, researchers could utilize one or more of these principles to design a viewpoint quality metric.

3. Viewpoint Quality Metrics

This section surveys all of the metrics that have been used as a viewpoint quality measurement to select the best camera placement. The metrics have been categorized into two sections. Section 3.2 surveys the viewpoint quality measures that are based on the geometry of the data. And Section 3.3 surveys the viewpoint quality measures that are based on the field data.

3.1. Notation

This section will define the notation used for these metrics. Table 1 comes from the in-depth survey from Bonaventura et al. [BFS*18] where they compared 22 different viewpoint metrics. The notation developed is based off of an information channel developed by Feixas et al. [FSG09]. The information channel is defined between a set of viewpoints V and a set of polygons Z of an object. For some polygon $z \in Z$ and some viewpoint $v \in V$, the projected area of polygon z from viewpoint v is denoted $a_z(v)$. Similarly, the projected area of the model Z from some viewpoint $v \in V$ is denoted $a_t(v)$.

Feixas et. al [FSG09] created a selection framework based off of their information channel. The information channel is defined by a matrix of conditional probabilities based on the ability to view polygons given some $v \in V$. Since the conditional probabilities represent the probability of viewing a particular polygon z from some viewpoint v, the information channel can also be considered a visibility channel. The information channel defines three elements:

- The conditional probability matrix, $p(Z|V)$, is made up of the surface area ratios $p(z|v)$, such that $p(z|v) = \frac{a_z(v)}{a_t(v)}$ and $\sum_{z \in Z} p(z|v) = 1$.
- The input distribution $p(V)$ represents the importance of each viewpoint within the set of views. The importance distribution is made up of elements $p(v) = \frac{a_t(v)}{\sum_{v \in V} a_t(v)}$.
- The output distribution $p(Z)$ represents the average projected area of polygon z and is made up of elements $p(z) = \sum_{v \in V} p(v)p(z|v)$.

Notation	Definition
z	polygon
Z	set of polygons
v	viewpoint
V	set of viewpoints
$a_z(v)$	projected area of polygon z from viewpoint v
$a_t(v)$	projected area of the model from viewpoint v
$vis_z(v)$	visibility of polygon z from viewpoint v (0 or 1)
N	number of polygons
R	number of pixels of the projected image
A_z	area of polygon z
A_t	total area of the model
$p(z\|v)$	conditional probability of z given v
$p(z)$	probability of z
$p(v\|z)$	conditional probability of v given z
$p(v)$	probability of v
$H(V)$	entropy of the set of viewpoints
$H(Z)$	entropy of the set of polygons
$H(V\|z)$	conditional entropy of the set of viewpoints given polygon z
$H(Z\|v)$	conditional entropy of the set of polygons given viewpoint v
$slength(v)$	silhouette length from viewpoint v
$\{h(\alpha)\}$	normalized silhouette curvature histogram
α	turning angle bin
a	turning angle between two consecutive pixels
A	set of turning angles
N_a	number of turning angles
$depth(v)$	normalized maximum depth of the scene from viewpoint v
$\{h(d)\}$	normalized histogram of depths
d	depth bin
D	set of depth bins
N_v	number of neighbors of v
$L(v)$	size of the compression of the depth image corresponding to viewpoint v
$L(v_i, v_j)$	size of the compression of the concatenation of the depth images corresponding to viewpoints v_i and v_j
K_i	curvature of vertex i
$\{h(b)\}$	normalized histogram of visible curvatures from viewpoint v
b	curvature bin
B	set of curvature bins
$S(x)$	saliency of vertex x

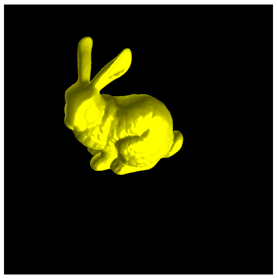
TABLE 1: Notation developed in the survey by Bonaventura et al. [BFS*18].

Work by Bonaventura et al. [BFS*18] and Secord et al. [SLF*11] have categorized the following 22 viewpoint quality measures into five different categories based on the calculations involved. The five categories of measurements are: area, silhouette, depth, stability, and surface curvature. This paper will follow the same naming conventions and categorizations for each measurement.

3.2. Geometry Based Quality Measures

This section will cover all automatic viewpoint selection metrics that are based on geometry, e.g. surface curvature, polygons, mesh saliency, etc. While some of these metrics are used in fields other than computer science, the majority of them have been utilized in determining camera position for image-based modeling. Few of these metrics have been applied to scientific data sets and all have been applied post-hoc.

3.2.1. Area. This section covers the measurements that involve the projected area of the polygons in relation to a particular viewpoint, including view area, ratio of visible area, and surface area entropy. These metrics are useful on datasets with highly varying polygons and when maximizing visible area is important. They are all relatively quick, unless they involve mutual information, and can be simply


(a) The best viewpoint of the Stanford Bunny.

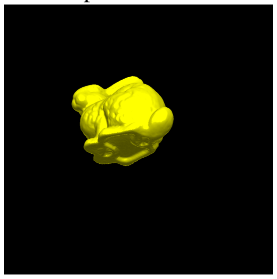
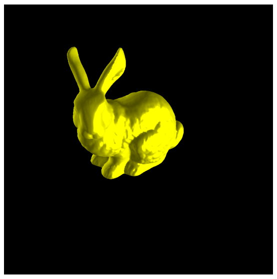

(b) The worst viewpoint of the Stanford Bunny.

Figure 2: The best and worst viewpoints of the Stanford Bunny determined by metric VQ_1.


(a) The best viewpoint of the Stanford Bunny.

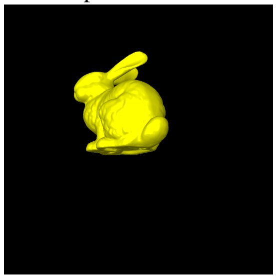

(b) The worst viewpoint of the Stanford Bunny.

Figure 3: The best and worst viewpoints of the Stanford Bunny determined by metric VQ_2.

implemented with the aid of graphics hardware [BDP99], [BDP00].

Number of Visible Triangles. Several of the first measurements on viewpoint selection came from Plemenos [Ple91] and were then expanded upon by Plemenos and Benayada [PB96]. The first measurement is based on the total number of visible triangles from some viewpoint. Their reasoning being that maximizing information means maximizing details, and the more triangles present, the more details associated with that view. This viewpoint quality measurement is defined as follows:

$$VQ_1(v) = \sum_{z \in Z} vis_v(z).$$

This measurement can be implemented differently depending on the chosen definition of visible. For most implementations, a polygon z is considered visible if any portion of it is viewable from the given v ($a_z(v) > 0$).

Unfortunately, this measurement has an obvious pitfall. By being based solely on the number of visible triangles, this measurement favors quantity over quality and could potentially choose views that contain a lot of polygons but little content. Figure 2 shows the best and worst viewpoints of the Stanford Bunny based on VQ_1.

Projected Area. Plemenos and Benayada [PB96] realized that their first measurement, $VQ_1(v)$, may not be an adequate measurement in some cases, and they should take into account the projected area of the polygons.

Their second measurement is simply the total visible area of the model from some viewpoint is defined as follows:

$$VQ_2(v) = a_t(v).$$

For this measurement, the higher the projected area of the model, the better the viewpoint.

This metric also has known pitfalls. In the worst case, this metric could potentially choose a viewpoint that contains a single, very large polygon. Further, by maximizing the visible area there is the risk of potentially maximizing the number of occlusions. Figure 3 shows the best and worst viewpoints of the Stanford Bunny using this metric.

Plemenos and Benayada. The next measurement from Plemenos and Benayada is a combination of $VQ_1(v)$ and $VQ_2(v)$, creating a viewpoint ratio based on both the total number of visible triangles and the total projected area. This measurement is expressed as follows:

$$VQ_3(v) = \frac{\sum_{z \in Z} \left\lceil \frac{a_z(v)}{a_z(v)+1} \right\rceil}{N} + \frac{\sum_{z \in Z} a_z(v)}{R}$$

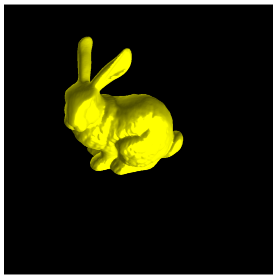
where N is the total number of polygons of the model ($N = |Z|$) and R is the resolution of the image (i.e. total number of pixels).

The best viewpoint will have the highest value, corresponding to a viewpoint that maximizes the number of visible triangles as well as the resolution of the rendered image. This is a quick and generic metric that should produce adequate results for most data sets.

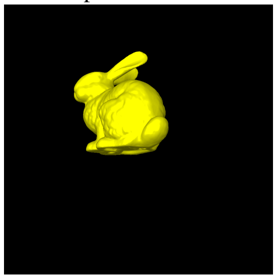
Figure 4 shows the best and worst viewpoints of the Stanford Bunny using this metric.

Visibility Ratio. Lastly, Plemenos and Benayada [PB96] measured the visibility ratio of the model given some viewpoint. Interestingly, their ratio involves the real surface area of some polygon z and not its projected area (i.e. the area of the triangle in World Space).

This measurement is defined as follows:



(a) The best viewpoint of the Stanford Bunny.



(b) The worst viewpoint of the Stanford Bunny.

Figure 4: The best and worst viewpoints of the Stanford Bunny determined by metrics VQ_3 and VQ_7.

$$VQ_4(v) = \frac{\sum_{z \in Z} vis_v(z) A_z}{A_t}$$

The higher the visibility ratio the better the viewpoint.

Notice that A_z and A_t are the real area of polygon z and the model, respectively, and are insensitive to the chosen viewpoint.

This metric has similar pitfalls and advantages as VQ_2.

Viewpoint Entropy. To determine the best, most representative viewpoint of an image, a new technique is used involving information theory called Viewpoint Entropy [VFSH01], [SFR*02].

Viewpoint Entropy is based off of Shannon Entropy [CT06], [Bla87a]. From a high level, Shannon Entropy determines the saliency of data by calculating how many bits are required to save the given data. The more bits that are required, then the more information that is present.

From a low level, Shannon Entropy is the summation of the negative log of the probability mass function of each possible data value:

$$S = -\sum_i P_i \log P_i$$

To calculate Viewpoint Entropy, Vazquez et al. [VFSH01] alter Shannon Entropy to take into account the projected area of the scene when centered at a particular viewpoint.

To define viewpoint entropy, let $a_z(v)$ be the projected area of polygon z from viewpoint v, let $a_t(v)$ be the total projected area of the model from viewpoint v, and let N be the total number of polygons of the model. Then, viewpoint entropy of a given view v is defined as follows:

$$VQ_5(v) = -\sum_{i=0}^{N} \frac{a_z(v)}{a_t(v)} \log \frac{a_z(v)}{a_t(v)}$$

The ratio $\frac{a_z(v)}{a_t(v)}$ represents the proportion of the projected area of each polygon. This ratio is also proportional to the cosine of the angle between the normal of the projected polygon $a_z(v)$ and the camera angle. Additionally, this ratio is inversely proportional to the squared distance from the camera to polygon. This means that $\frac{a_z(v)}{a_t(v)}$ will be higher when the polygon is seen from a better angle and at a closer distance.

Viewpoint entropy can be rewritten in terms of conditional probabilities, since it measures the conditional entropy of Z given some v. Using conditional probabilities, viewpoint entropy can be defined as follows:

$$VQ_5(v) = H(Z|v) = -\sum_{z \in Z} p(z|v) \log p(z|v).$$

Given this definition, the best camera position for a scene is the view that has the highest viewpoint entropy and thus the highest information content. This metric will work best on data sets with varying polygonal size, since larger polygons are penalized in comparison to smaller polygons.

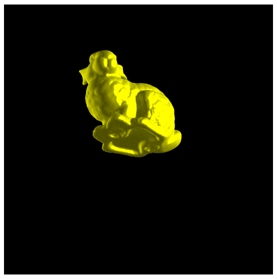
Polonsky et al. [PPB*05] also considered viewpoint entropy and a handful of the other measurements as *view descriptors*. But instead of assigning values to primitive elements of the model (e.g. vertices, faces, edges), importance is assigned to segments and connected components.

I$_2$. The measurement I$_2$ is a normalization of VQ_5 and has been used in neuroscience by DeWeese and Meister [DM99] to quantify the information content in the brain in terms of stimuli and response. Bonaventura et al. [BFS11] also applied this measure to selecting the best viewpoint. I$_2$ is defined as follows:

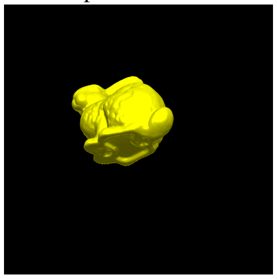
$$\begin{aligned} VQ_6(v) = I_2(v; Z) &= H(Z) - H(Z|v) \\ &= H(Z) - VQ_5(v) \\ &= -\sum_{z \in Z} p(z) \log p(z) + \sum_{z \in Z} p(z|v) \log p(z|v). \end{aligned}$$

$H(Z)$ represents the entropy of the polygons and is constant for every viewpoint. Notice that I$_2$ is based off of viewpoint entropy, $H(Z|v)$, and subsequently has the same behavior, but in this case, the higher the viewpoint entropy then the lower the value of I$_2$ will be. And unlike viewpoint entropy, which grows to infinity for finer and finer mesh resolutions, the normalization within I$_2$ makes its behavior more stable.

Viewpoint Kullback-Leibler Distance (VKL). The Kullback-Leibler distance was applied by Sbert et al. [SPFG05] as a viewpoint quality measurement between the normalized distribution of the real areas of the polygons, and the normalized distribution of the projected area of the polygons from viewpoint v. The VKL distance is defined as follows:



(a) The best viewpoint of the Stanford Bunny.



(b) The worst viewpoint of the Stanford Bunny.

Figure 5: The best and worst viewpoints of the Stanford Bunny determined by metric VQ_8.

$$VQ_7(v) = \sum_{z \in Z} \frac{a_z(v)}{a_t(v)} \log \frac{\frac{a_z(v)}{a_t(v)}}{\frac{A_z}{A_t}}.$$

Notice that the best viewpoint, which corresponds to the minimum value, happens when the distribution of projected areas is equal to the distribution of real areas.

Figure 4 shows the best and worst viewpoints of the Stanford Bunny using this metric.

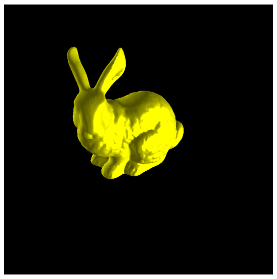
Viewpoint Mutual Information ($\mathbf{I_1}$). Feixas et al. [FSG09] introduces this viewpoint selection method that quantifies the degree of correlation between the viewpoints and set of polygons. I_1 is expressed as follows:

$$VQ_8(v) = I_1(v; Z) = \sum_{z \in Z} p(z|v) \log \frac{p(z|v)}{p(z)}.$$

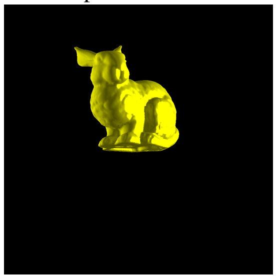
High values of I_1 correspond to representative views for certain areas of the model. Meaning that the respective image is highly coupled with the polygons from that viewpoint, and certain regions of the model can only be seen from this or few other viewpoints. Alternatively, low values of I_1 correspond to the most representative views of the entire model, i.e., views that capture the most polygons in a balanced way. This measure has been used in neuroscience to capture the correlation between stimuli and brain responses [DM99].

Note that mutual information is a time intensive calculation, making this metric, as well as the following metric, potentially less desirable than the 7 previous metrics, which perform much more quickly.

Figure 5 shows the best and worst viewpoints of the Stanford Bunny using this metric.



(a) The best viewpoint of the Stanford Bunny.



(b) The worst viewpoint of the Stanford Bunny.

Figure 6: The best and worst viewpoints of the Stanford Bunny determined by metric VQ_9.

Information $\mathbf{I_3}$. Again from neuroscience, Butts [But03] presents a metric to quantify the information associated with a stimulus using mutual information. Whereas Bonaventura et al. [BFS11] proposed this metric as a viewpoint quality measure. I_3 is expressed as follows:

$$VQ_9(v) = I_3(v; Z) = \sum_{z \in Z} p(z|v) I_2(V; z)$$

such that

$$\begin{aligned} I_2(V; z) =& H(V) - H(V|z) \\ =& -\sum_{v \in V} p(v) \log p(v) + \sum_{v \in V} p(v|z) \log p(v|z) \end{aligned}$$

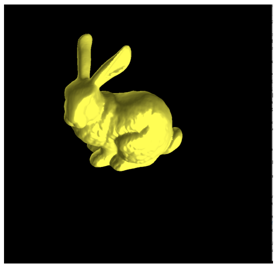
where $I_2(V; z)$ is comprised of the entropy of the viewpoints, and the conditional entropy of the viewpoints for polygon z. A high I_3 means a high $I_2(V; z)$ value and corresponds to the view that sees the highest number of "maximally informative polygons." [BFS*18]

Figure 6 shows the best and worst viewpoints of the Stanford Bunny using this metric.

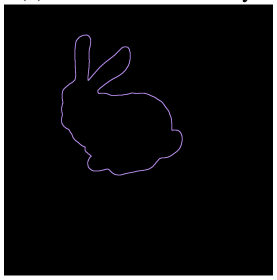
3.2.2. Silhouette.

3.2.2. Silhouette. This section surveys metrics that utilize a model's silhouette. The silhouette, or *occluding contour*, of an object is a view-dependent metric. Simply, the silhouettes of an object are the edges that are created if the object were to be represented as a single color on a plain background, as shown in Figure 7.

In computer science, silhouettes have most commonly been used for image recognition. For scientific data, silhouettes are best used on datasets with lots of occlusions. Since silhouettes are a view-dependent metric, a model with significant occlusions, either in number or size, will result



(a) The Stanford Bunny.



(b) The corresponding, view-dependent silhouette of Image 7a.

Figure 7: An example of a silhouette of a model.

in silhouettes that are specific to certain viewpoints. The following metrics utilize a model's silhouette to determine the best viewpoint.

Silhouette Length. The use of an object's silhouette as a goodness measure was presented by Polonsky et al. [PPB*05]. From the given viewpoint, the silhouette of the model is calculated by counting the number of pixels that belong to the silhouette, or edge, of the object. The silhouette length is defined as follows:

$$VQ_{10}(v) = slength(v).$$

In the cases where there are multiple connected components and, subsequently, multiple silhouettes, the silhouette of each component is combined for a total sum of the silhouette lengths. The maximum silhouette is the longest silhouette and is associated with the best viewpoint.

Silhouette Entropy. Page et al. [PKS*03] was the first to combine silhouettes and entropy, and it was Polonsky et al. [PPB*05] who first identified their approach as a metric for viewpoints. The entropy of a curve is the entropy of the curvature distribution. The histogram for the silhouette curvature distribution is computed using the angles between the pixels that make up the silhouette. In the discrete case, the turning angles range from $\frac{-\pi}{2}$ to $\frac{\pi}{2}$ with a step of $\frac{\pi}{4}$, and entropy is calculated for all turning angles between adjacent silhouettes. Silhouette entropy is defined as follows:

$$VQ_{11}(v) = -\sum_{\alpha=-\frac{\pi}{2}}^{\frac{\pi}{2}} h(\alpha)\log h(\alpha),$$

where $h(\alpha)$ is the normalized silhouette curvature histogram and α is the bin for a particular turning angle. The highest silhouette entropy corresponds to the best viewpoint.

Silhouette Curvature and Silhouette Curvature Extrema. Vieira et al. [VBP*09] introduced a new metric, based on the work by Felldman et al. [FS05], where the integral curve of the silhouette is calculated. Silhouette curvature is defined as follows:

$$VQ_{12}(v) = \frac{\sum_{c\in C}\frac{|c|}{\frac{\pi}{2}}}{N_c},$$

where c is the turning angle between two consecutive pixels, C is the set of turning angles, and N_c is the number of turning angles (s.t. $|N_c| = slength(v)$). The highest silhouette curvature corresponds to the best viewpoint.

Secord et al. [SLF*11] slightly alters silhouette curvature by enhancing the impact of the turning angles, this will emphasize any extreme curvatures present in the silhouette. Silhouette curvature extrema is expressed as

$$VQ_{13}(v) = \frac{\sum_{c\in C}(\frac{|c|}{\frac{\pi}{2}})^2}{N_c}.$$

Similarly, the higher the silhouette curvature extrema, the better the viewpoint.

Note that for both curvature and curvature extrema, intersecting silhouettes create T-junctions that can create high curvatures and create false positives when searching for the best viewpoint.

3.2.3. Depth. This section reviews the metrics that involve the model depth. Depth is a natural metric for choosing the best viewpoint because depth, and portraying depth, is a key component to three dimensional renderings. For certain data sets, such as terrain, taking into account the depth of the model is a necessity. For example, if we were to apply the area metrics to a terrain model, the best viewpoint will most likely be an overhead view that makes the terrain appear flat.

Stoev and Straber. Stoev and Staber [SS02] realized that the area metrics perform poorly on terrain data sets and introduced a metric that selects the viewpoint that best maximizes both the projected area and the projected depth.

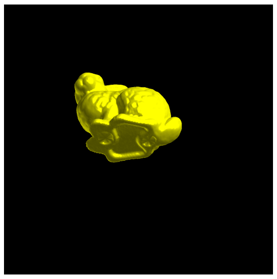
The Stoev and Straber metric is defined as follows:

$$VQ_{14}(v) = \alpha p(v) + \beta d(v) + \gamma(1 - |d(v) - p(v)|),$$

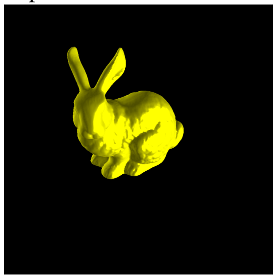
where $p(v)$ is the normalized projected area of the model from some viewpoint v, and $d(v)$ is the normalized maximum depth of the model from some viewpoint v. For general purposes, the authors set $\alpha = \beta = \gamma = \frac{1}{3}$. And for terrain models the authors recommend setting $\alpha = \beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$.

For this metric, the highest value will correlate to the best viewpoint. This metric is best used when projected area as well as depth are key components of the model, such as with terrain data sets. This metric is adaptable and allows for some user input in terms of prioritizing depth, area, or both.

Maximum Depth. Secord et al. [SLF*11], inspired by Stoev and Straber, also considered depth when creating a



(a) The best viewpoint based on maximum depth, VQ_{16}.



(b) The worst viewpoint based on maximum depth, VQ_{16}.

Figure 8: The best and worst viewpoints of the Stanford Bunny based on the metric Maximum Depth.

viewpoint metric. Their first metric is simply the maximum depth of the model. Maximum depth is defined as follows:

$$VQ_{15}(v) = depth(v),$$

where $depth(v)$ is the maximum depth of the model from some viewpoint v.

For this metric, the best viewpoint will have the greatest depth. This metric should work well for terrain datasets, but could perform poorly on image-based models, as shown in Figure 8.

Depth Distribution. Secord et al. [SLF*11] also took into account the depth distribution of the image for each viewpoint. Their metric maximizes the range of depths and chooses the camera placement that has the most equally distributed view of depths.

Depth distribution is defined as follows:

$$VQ_{16}(v) = 1 - \sum_{d \in D} h(d)^2,$$

where d is a depth bin, D is the set of bins, and $h(d)$ is the normalized histogram of depths. Figure 9 is an example of a normalized historgram and corresponds to Figure 7a.

The best viewpoint will have the most even distribution of depths. Hence, this metric will work well for most 3-dimensional data sets.

3.2.4. Stability. The metrics in this subsection involve visual stability, which is determined by analyzing a neighborhood of surrounding views (defined by a threshold). If the difference between the view and its neighboring views are large, then that view is said to be unstable, conversely, if the difference is small, then that view is considered stable.

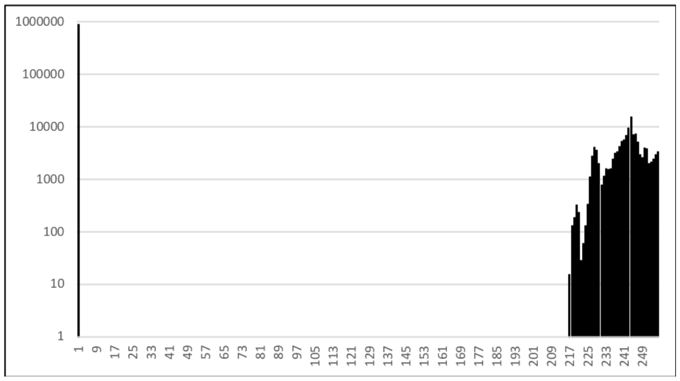


Figure 9: The normalized depth histogram corresponding to Figure 7a. The Depth Distribution metric will favor viewpoints whose histograms are evenly distributed with few peaks or valleys. This analysis used 256 bins for the 1M pixels of the image.

From user studies, a stable viewpoint has been shown to be more visually appealing, whereas an unstable viewpoint is a good starting point for post-hoc exploration since the user can see a large change in the model with only slight changes to the view.

Instability. Bordoloi and Shen [BS05] were the first to consider the similarities between viewpoints for volume rendering. To find similar or dissimilar images, they use the projected area distributions associated with each viewpoint and compute the distance using the Jensen-Shannon divergence measure [BR82]. They considered the Kullback-Leibler difference [Bla87b] as well, but two images with differing occlusions cannot be compared with that measure.

Feixas et al. [FSG09] expanded on this research and used instability as a metric for image-based rendering. And Lin [Lin91] showed that the Jensen-Shannon divergence of projected area distributions used in stability calculations can be expressed in terms of Shannon Entropy. Instability is calculated as follows:

$$VQ_{17}(v) = \frac{1}{N_v} \sum_{j=1}^{N_v} D(v, v_j),$$

where v_j is a neighboring view of v, N_v is the number of neighbors of v, and $D(v, v_j)$ is the Jensen-Shannon divergence of the projected area distributions. $D(v, v_j)$ is defined as follows:

$$D(v, v_j) = JS\left(\frac{p(v)}{p(v) + p(v_j)}, \frac{p(v_j)}{p(v) + p(v_j)}; p(Z|v), p(Z|v_j)\right),$$

where $p(Z|v)$ and $p(Z|v_j)$ are the distributions with weights $p(v)/(p(v) + p(v_j))$ and $p(v_j)/(p(v) + p(v_j))$, respectively.

For this metric, the lowest instability is the best viewpoint. This metric is suitable for data producers who want a viewpoint that will appeal to most users.

Depth-based Visual Stability. Vazquez et al. [VFSH01] computes stability using the corresponding depth images from every viewpoint. To determine the similarity between two depth images, the Normalized Compression Distance (NCD) is utilized. NCD is defined as follows:

$$NCD(v_i, v_j) = \frac{L(v_i v_j) - min\{L(v_i), L(v_j)\}}{max\{L(v_i), L(v_j)\}},$$

where $L(v_i)$ and $L(v_j)$ are the sizes of the compressed depth images for viewpoints v_i and v_j respectively, and $L(v_i v_j)$ is the sized of the compressed concatenation of the depth images for views v_i and v_j.

A view is considered similar to another if their corresponding NCD score is less than a given threshold. The best viewpoint will be the one that has the largest number of similar views. Hence, the depth-based visual stability metric is defined as follows:

$$VQ_{18}(v) = \textit{number of similar views to v}.$$

Again, this metric is useful when user appeal/preference is aesthetically important.

3.2.5. Surface Curvature. The metrics in this section analyze the curvature of the model's surface in order to determine the best viewpoint. Intuitively, the curvature of a surface is the amount of curve the surface deviates from being flat.

Curvature Entropy. Page et al. [PKS*03], interested in shape analysis, first proposed calculating the entropy of the Gaussian curvature distribution over the entire surface of the object. Polonsky et al. [PPB*05] built off of this work to develop a metric that calculates the entropy of the curvature distribution over the visible portion of the object's surface. With this metric, the best viewpoint will maximize the projection of unique curvature present in the model.

The curvature of vertex x is estimated by the standard angle deficit approximation:

$$K_x = 2\pi \sum_j \phi_j,$$

where angle ϕ_j is the wedge subtended by the edges of a triangle whose corner is at vertex x.

The curvature entropy is defined as follows:

$$VQ_{19}(v) = -\sum_{b \in B} h(b) \log h(b),$$

where b represents a curvature bin, B is the set of curvature bins, and $h(b)$ is the normalized histogram of visible curvatures from viewpoint v.

The best viewpoint will have the highest curvature entropy. This metric is fitting for data sets that place importance on depth and specifically the angles of the peaks and valley's that make up the model's visible surface.

Visible Saliency. Lee et al. [LVJ05] present a metric for computing the mesh saliency of a 3D object. This is based on the center-surround method from Itti et al. [IKN98] which computed the saliency for 2D images.

To calculate visible saliency, the curvature at every vertex is calculated using the strategy presented by Taubin [Tau95]. Next, the saliency, $S(x)$, of each vertex, x, is calculated using the Gaussian-weighted average of the mean curvature, defined as follows:

$$S(x) = |G(C(x), \sigma) - G(C(x), 2\sigma)|,$$

where $G(C(x), \sigma)$ is the Gaussian-weighted average of the mean curvature and σ is Gaussian's standard deviation. Multiple saliency maps are created by varying σ. The final saliency is the aggregate of the saliency maps with a non-linear normalization. Visible saliency is defined as follows:

$$VQ_{20}(v) = \sum_{x \in X} S(x),$$

with X being the set of visible vertices and $S(X)$ being the saliency of vertex x. Figure 10 shows how mesh saliency is calculated. The best viewpoint will be the one with the highest visible salience value.

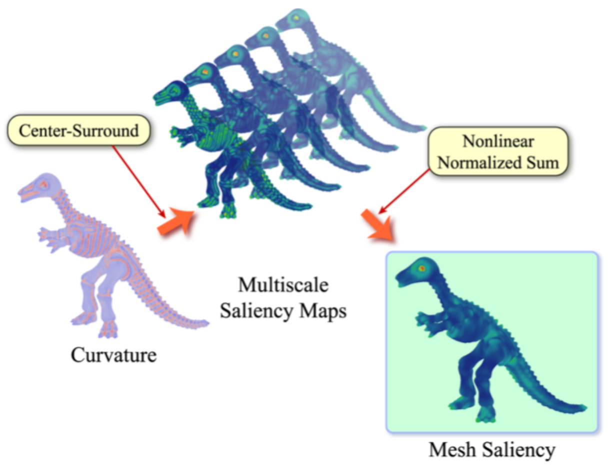
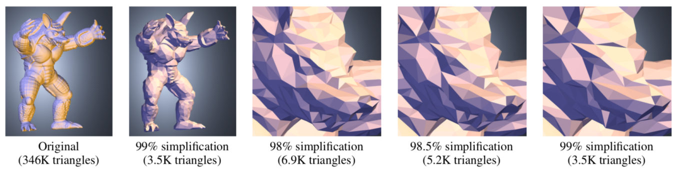


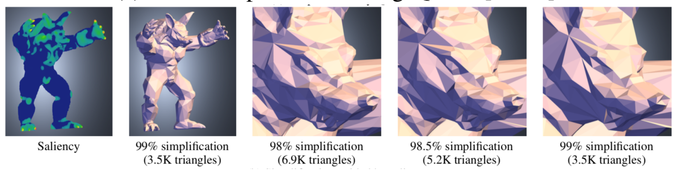
Figure 10: To compute visible saliency, the mean curvature is computed for each vertex. The vertex saliency is then computed as the difference between mean curvatures filtered with a narrow and broad Gaussian. Multiple saliency maps are computed by varying σ, the Gaussian standard deviation. Lastly, the saliency of the viewpoint is the aggregate of all the saliency maps using a non-linear normalization. Taken from Lee et al. [LVJ05].

Sokolov and Plemenos [SP05] also implemented this metric, but for curvature they used the standard angle deficit approximation K_x, as seen in VQ_{19}.

The authors believed that a curvature peak within a flat region is as important as a flat region in the middle of dense peaks. This metric aims to quantify the variations present in the geometry, the more intense the variations, the higher the saliency. Subsequently, zero saliency will correspond to a region with uniform intensity, such as a sphere. Mesh saliency and the produced saliency map can



(a) Mesh simplification using QSlim [GZ05].



(b) Mesh simplification guided by the saliency map.

Figure 11: Taken from [LVJ05], Lee et al. visually compared the QSlim mesh simplification to their saliency-guided mesh simplification. The mesh saliency map produced from this metric guides the mesh simplification in order to keep the highly salient aspects of the object intact.

be a preservation guide when applying other visualization operations, such as mesh simplification, shown in Figure 11.

Projected Saliency. Feixas et al. [FSG09] apply the idea of mesh saliency to individual polygons. The saliency of a polygon is defined as the average dissimilarity between this polygon and its neighbors. This metric is based on mutual information, similar to metric VMI from Section 3.2.1.

The saliency, $S(z)$, of polygon z is defined as:

$$S(z) = \frac{1}{N_z} \sum_{j=1}^{N_z} D(z, z_j),$$

where z_j is a neighboring polygon of z, N_z is the set of heights of z, and

$$D(z, z_j) = JS(\frac{p(z)}{p(z)+p(z_j)}, \frac{p(z_j)}{p(z)+p(z_j)}; p(V|z), p(V|z_j)),$$

is the Jensen-Shannon divergence between the distributions $p(V|z)$ and $p(V|z_j)$ with weights $p(z)/(p(z)+p(z_j))$ and $p(z_j)/(p(z)+p(z_j))$, respectively.

The visible saliency is defined as follows:

$$VQ_{21}(v) = \sum_{z \in Z} S(z) p(v|z),$$

where the best viewpoint will have the highest visible saliency.

This metric takes into account the saliency of individual polygons rather than the curvature of the mesh surface, meaning it will do well on amorphous geometries comprised of locally unique polygons. But, much like the other polygonal metrics that use mutual information, this metric is likely to have a long execution time.

Saliency-based EVMI. Feixas et al. [FSG09] extends VMI from Section 3.2.1 to include a weighted importance factor. Saliency-based EVMI is defined as follows:

$$VQ_{22}(v) = \sum_{z \in Z} p(z|v) \log \frac{p(z|v)}{p'(z)},$$

with $p'(z)$ defined as:

$$p'(z) = \frac{p(z)i(z)}{\sum_{z \in Z} p(z)i(z)},$$

and $i(z)$ is the importance of polygon z.

Serin et al. [SSB13] alter this metric, redefining $i(z)$ as the curvature of polygon z and in place of $p(z)$ they use the total area of polygon z, a_z.

The best viewpoint will correspond to the minimum value. This metric will do well with geometries comprised of differing polygons, but will have a long execution time, similar to the other metrics based on mutual information.

3.3. Data Driven Quality Measures

This section will cover all automatic viewpoint selection metrics that are based on data. Metrics in this category would most likely be applied to scientific data sets, such as scalar fields and volumetric data that lie on a regular mesh.

There has been little research done to define data-driven metrics that will determine the best viewpoint of a regular grid. The solutions that have been proposed involve information theory. While this survey only covers entropy informed camera placement, Wang and Shen [WS11] survey the use of information theory in scientific visualization.

3.3.1. Entropy. This section will cover all metrics that utilize entropy to determine the best viewpoint.

Viewpoint Entropy. The first metric is to utilize viewpoint entropy established by Vazquez et al. [VFSH03], [VFSH01]. Viewpoint entropy would maximize the visible entropy of each face. Unfortunately, a regular mesh has at most six faces.

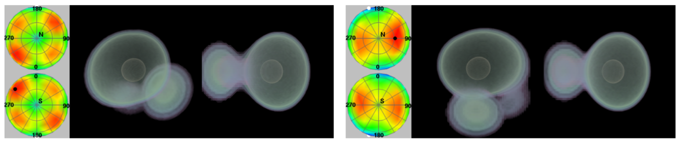
This is an easily implementable and quick metric that could be applied to all 3D regular datasets. A significant issue with this metric — and all of the geometry-based metrics — is that it assumes the surfaces have zero thickness.

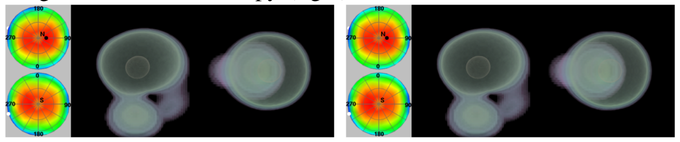
Isosurface Entropy. Takahashi et al. [TFTN05] applied viewpoint entropy to volumetric data, utilizing the inherent geometry that is based on the transfer function. Let $p_i (i = 0, \ldots, n-1)$ be the set of scalar values that has been uniformly sampled from the entire data set. This set of scalar values will be used to create n individual isosurfaces, $I_i (i = 0, \ldots, n-1)$.

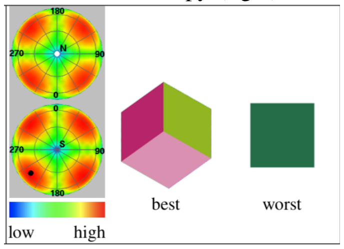
The viewpoint entropy of an individual isosurface, $E_i(v)$, is defined as follows:

$$E_i(v) = \frac{1}{\log(m_i+1)} \sum_{j=0}^{m_i} \frac{a_{ij}}{R} \log \frac{a_{ij}}{R},$$

where $a_{ij} (j = 0, \ldots, m_i)$ is the j^{th} visible face of the i^{th} isosurface, R is the resolution of the viewpoint (i.e. pixel


(a) The best and worst images based on isosurface entropy (left) and weighted isosurface entropy (right).


(b) The best and worst images based on interval volume entropy (left) and weighted interval volume entropy (right).



(c) The key for Figure 12.

Figure 12: The viewpoint entropy distributions as well as the best and worst images based on the isosurface (12a) and interval volume (12b) metrics, and respective key (12c). Taken from [TFTN05].

resolution), m_i is the total number of visible faces of the isosurface, and $\frac{1}{\log m_i + 1}$ normalizes the value.

The viewpoint entropy of the entire volumetric data set is the average of the individual isosurface entropies and is defined as follows:

$$VQ_{23}(v) = \frac{1}{n}\sum_{i=0}^{n-1} E_i(v).$$

The best viewpoint will have the highest isosurface entropy.

A downside of this metric is that it uses the isosurfaces individually and does not take into account occlusions, as shown in Figure 12a.

Weighted Isosurface Entropy. Wanting to improve their previous metric to account for occlusions, Takahashi et al. [TFTN05] assign different weights to the individual isosurfaces, helping to accentuate certain features and choose views with fewer occlusions. The weight, λ_i, for the i^{th} isosurface is computed using the transfer function. Transfer functions can be designed to emphasize the inherent geometry present in the scalar field [TTF04], [WS04].

Denote the transfer function of a scalar value s as $TF(s)$. Then λ_i is defined as follows:

$$\lambda_i = TF(p_i),$$

where p_i is the isovalue for isosurface, I_i.

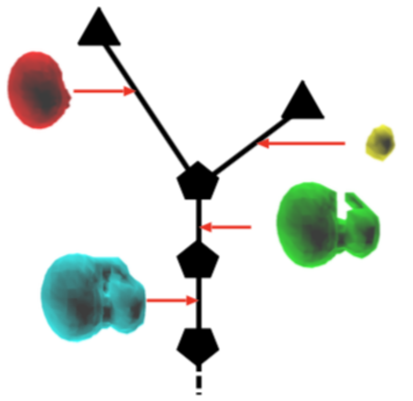


Figure 13: An example of a contour tree and the respective interval volumes associated with sections of the tree. Taken from [TFTN05].

Then the weighted isosurface entropy is defined as follows:

$$VQ_{24}(v) = \sum_{i=0}^{n-1} \frac{\lambda_i}{L} E_i(v),$$

where $L = \sum_{i=0}^{n-1} \lambda_i$.

The best viewpoint will correspond to the highest entropy.

Adding these weights resulted in fewer occlusions, but still suffers from overlaps as shown in Figure 12a. They found this was the case for other volumetric data sets as well.

Interval Volume Entropy.

Takahashi et al. [TFTN05] hypothesized that an isosurface-based metric could not provide satisfactory results, even when individual results are weighted, due to three reasons:

- Universally sampling the data for isovalues does not precisely reflect all the shapes present in the data, and it's possible the most interesting isovalue will not be selected.
- The approach cannot distinguish between connected components of a single isosurface, meaning they cannot assign different weights to disjoint components.
- The isosurface is neglecting the overall thickness present in the data.

The authors shift their focus towards *interval volumes*. Interval volumes are defined as a subvolume composed of isosurface within a range of scalar field values. The authors use an *interval volume decomposer* (IVD) [TFT05] to represent an interval volume as a contour tree [BPS97], which tracks the topological transitions of isofields with respect to the scalar field as shown in Figure 13. Once the data has been decomposed, entropy can be calculated. The viewpoint entropy of an individual interval volume $E_i^v(v)$ is defined as follows:

$$E_i^v(v) = \frac{1}{\log(m_i+1)} \sum_{j=0}^{m_i} \frac{a_{ij}}{R} \log \frac{a_{ij}}{R},$$

where a_{ij} is the j^{th} face of the i^{th} interval volume $V_i(i = 0, \ldots, n)$, and R is the total area of the screen.

The interval volume entropy of the entire volumetric data set is the average of the individual isovolume entropies and is defined as follows:

$$VQ_{25}(v) = \frac{1}{n} \sum_{i=0}^{n-1} E_i^v(v).$$

Using this metric, the best viewpoint will correspond to the highest entropy.

By using interval volumes, the authors notice that the thickness of the data is more accurately portrayed as shown in Figure 12b.

Weighted Interval Volume Entropy.

Again, Takahashi et al. [TFTN05] build off the previous metric by adding weights to the interval volumes. The weights are based on a multidimensional transfer function. Figure 14 shows how using a multi-dimensional transfer function enhances the internal structure of the volume compared to a single-dimensional transfer function.

Let k_i be the number of voxels present in V_i and let $t_{ij}(j = 0, \ldots, k_i - 1)$ be the opacity value associated with the j^{th} voxel of the i^{th} interval volume. Then λ_i is defined as follows:

$$\lambda_i = \frac{1}{k_i} \sum_{j=0}^{k_i-1} t_{ij}.$$

Then the weighted interval volume entropy is defined as follows:

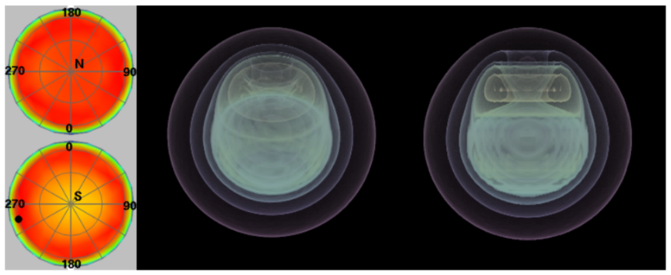
$$VQ_{26}(v) = \sum_{i=0}^{n-1} \frac{\lambda_i}{L} E_i^v(v),$$

where $L = \sum_{i=0}^{n-1} \lambda_i$.

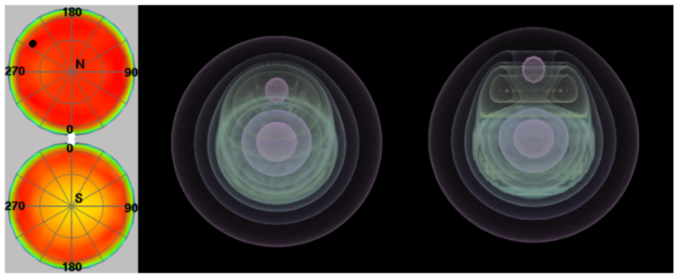
The best viewpoint will have the highest entropy.

The previous two metrics are applicable when the thickness of the data is important. And though Figure 12b shows only a modest difference between the two metrics, the use of a multi-dimensional transfer function can be critical to enhancing interior structures as shown in Figure 14, which is only utilized in the weighted interval volume metric.

As a final note, for extremely large data sets, these metrics are computationally heavy with potentially high communication costs. That said, utilizing software from Section 4 can help mitigate these performance costs.



(a) Weighted interval volume using a single-dimensional transfer function.



(b) Weighted interval volume using a multi-dimensional transfer function.

Figure 14: Viewpoint entropy distributions and the best and worst views of the simulation based on the weighted interval volume metric using a single-dimensional transfer function (14a) and a multi-dimensional transfer function (14b). By using a multi-dimensional transfer function the inner structures are emphasized. Taken from [TFTN05].

4. In Situ Analysis and Visualization Software

This section will survey the libraries and software that allow for in situ visualization and analysis. When applicable, for each software examined we determine how the default camera position is chosen.

ADIOS. The Adaptable IO System (ADIOS) [LLT*14] is a simple and flexible I/O middleware library that allows users to describe, write, read, or move data in situ. Using XML files as input, ADIOS easily lets users change how their I/O is handled. ADIOS has been shown to be scalable, portable, and efficient on supercomputers.

VTK-m. VTK-m [MSU*16] is a many core implementation of the Visualization Tool Kit (VTK) [SML06]. VTK is an open-source software that is used for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and 2D plotting. VTK-m has been used to rewrite visualization algorithms using data parallel primitives, allowing a single implementation to run efficiently on emerging architectures [LHK*16], [LMC*17], [LBMC16], [KCK*16], [LLCC17]. The VTK-m library plays a critical role in many of the following software implementations.

Ascent. Developed by Larsen et al. [LAA*17], Ascent is a multi-institutional project funded by the Exascale Computing Project (ECP) [Mes17]. A later implementation of Strawman [LBC*15], [HLB16], Ascent is a flyweight infrastructure that allows for in situ analysis and visualization of scientific datasets. Ascent utilizes VTK-m [MSU*16] for shared-memory parallelism using data-parallel primitives that are efficient and hardware agnostic. The goal of Ascent is three-fold:

- Provide in situ analysis and visualization for modern and emerging supercomputing architectures.
- Be a flyweight infrastructure with a simple interface, minimal dependencies on outside software, and minimal processing overheads when it comes to memory and copying data.
- Interoperability with other software. Ascent can support software other than VTK-m (such as R [RC14]). But it is up to the user to build a bridge for the data models to or from VTK-m.

Ascent also comes with several built-in scientific simulations.

In Ascent, unless otherwise specified, camera placement is based on the magnitude of the extents. This guarantees the data is in frame and is a canonical view for 3D datasets.

Cinema. Cinema [AJO*14] is an open source software that provides a unique approach to in situ and post-processing analysis and visualization of large scale scientific data. Unlike the other software that will typically only save a single image, Cinema saves a set of images in what the authors call a Cinema database. Using this database of images, the user can explore data at a fraction of the storage and I/O costs of saving the entire time slice. Beyond exploring data, Cinema provides sophisticated analysis and visualization routines that can be applied to the database. And now databases can now be composed of a range of metadata, such as run parameters, output variables, grids, or any other type of data that can written to disk.

Cinema has been integrated as an export option for VisIt/Libsim, Paraview/Catalyst, and Ascent. For this software, discussing camera placement does not make sense as this approach saves data from many viewpoints.

The components of the Cinema ecosystem are shown in Figure 15.

VisIt/Libsim. VisIt [CBW*12] is a free software for parallel visualization and analysis. VisIt allows users to generate visualizations of scientific data, animate through time, manipulate the data, and save images or viewpoint animations. Libsim [WFM11] is a tool in VisIt that facilitates in situ visualization. This allows users to explore the data interactively as the simulation is running, in addition to some debugging and simulation steering capabilities. Unless otherwise specified, the camera view defaults to a flat, 2D representation with the origin in the lower left corner.

ParaView/Catalyst. Paraview [AGL05] is an open source, data analysis and visualization application that utilizes VTK [SML06] under the hood. Providing both qualitative and quantitative techniques that users can perform either interactively or programmatically. Paraview was designed to be able to analyze extremely large datasets using distributed memory resources. Catalyst [ABG*15] is a library within Paraview that allows for in situ visualization and analysis.

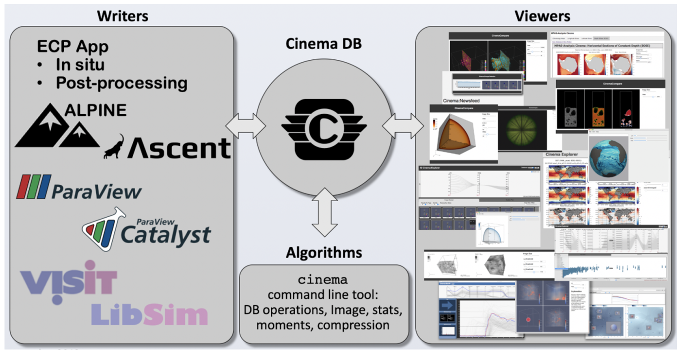


Figure 15: The Cinema ecosystem is composed of databases, algorithms, writers, and viewers. Cinema databases have been integrated as an export option for a number of state-of-the-art in situ analysis and visualization software, or writers. Cinema can employ sophisticated quantitative and qualitative algorithms on the database. The output can then be visualized using the Cinema viewer either post-hoc or in situ. Taken from [DLANL14].

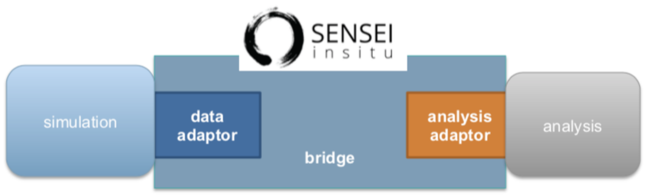


Figure 16: SENSEI is a generic interface that provides a bridge to many technologies. With a simple API, the user can decide which analysis routine to run. In other words, the user can "write once and run every where." Taken from [CLR*19].

Unless specified, the Paraview/Catalyst camera placement defaults to a rendering with the origin in the lower left corner.

SENSEI. SENSEI [AWW*16] is a generic in situ interface, providing the bridge from the simulation to a number of different visualization and analysis software, I/O file types, as well as in-transit data movement and analysis, as shown in Figure 16. The bridge takes the simulation data and using the data adaptor, exposes the simulation data structures, this is then passed to the analysis routines on the back-end via the analysis adaptor. With SENSEI, a user can can instrument their simulation code to the SENSEI API and then be able to utilize any of the in situ infrastructures that SENSEI currently supports (Ascent, VisIt/Libsim, Paraview/Catalyst, etc.). Additionally, a user can develop an in situ method using the SENSEI API with little modification to the simulation code.

For camera placement, unless otherwise specified, placement defaults to what the analysis routine dictates.

Overview. The aforementioned software are the main

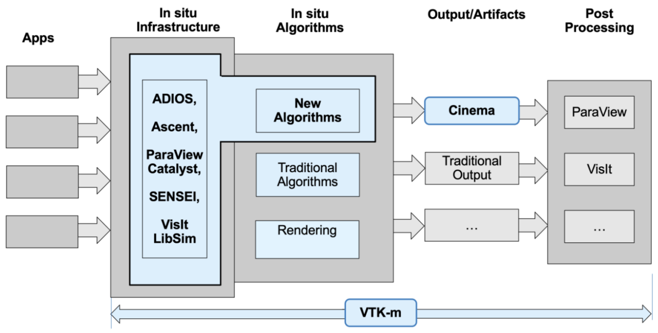


Figure 17: The current state-of-the-art for in situ analysis and visualization software and libraries. This diagram depicts the workflow from the simulation, to the in situ application, I/O routines, and then post processing. Taken from [DLANL14].

state-of-the-art in situ analysis and visualization applications and libraries being used today on large-scale simulations. Figure 17 shows where they fall within the in situ visualization and analysis workflow. Notice that VTK-m is heavily utilized among all these applications, proving its usefulness and necessity as supercomputers push for exascale.

5. Use Cases

In this section we look at recent works that apply in situ visualization to large-scale simulations or works that save out images. For each use case, we describe their method, and if applicable, how camera placement for rendering is determined.

Combustion.

Yu et al. [YWG*10] provide an early result within the in situ paradigm by proving the feasibility of in situ visualization at scale. Their in situ visualization produces high quality renderings of a large-scale turbulent-combustion simulation using S3D, a Sandia DNS (direct numerical simulation) solver. This particular simulation produces volume and particle data, so their work performs both highly parallel volume rendering and particle rendering in situ with manageable simulation strain (ranging from 2-36%). They performed their experiments using various configurations, with their largest run having 41 million particles, 600 million cells, executed on 15,360 cores. The camera position and transfer function for their renderings are user-specified, with the default being based on a small set of sample runs.

As analysis routines become more complex, they begin to take longer to execute and as a consequence, stall the simulation. A way to mitigate this is to transfer the data to secondary compute resources to perform the analysis, allowing the simulation to resume. But transferring data in itself can be prohibitive depending on the amount of data and desired configuration, whereas in situ analysis will have all the data locally. Work from Bennett et al. [BAB*12] present a hybrid method that utilizes both in situ and in transit data analytics to offload data that as been reduced or transformed in situ to secondary compute resources. Applying their method to a combustion simulation using S3D with a volume of 1 billion cells using 4,896 cores, their approach had an average wall time of 16 seconds per time step. They compare their hybrid methods to purely in situ algorithms, and show that, for the most part, their approach produces less or equitable strain on the simulation. Their hybrid method allows them to track, identify, downsample, and visualize features with feasible overhead. Much like their previous work [YWG*10], their camera placement is based on previous small scale runs.

Tokamak.

Work by Pugmire et al. [PKC*16] shows that near-real-time visualization and analysis is possible within a distributed workflow. This work lies in the cross-hairs of simulation monitoring and steering, and in situ visualization and analysis. There has been a lot of effort into developing steering and monitoring techniques for visualizing data across a network [Bet00b], [Bet00a], [BSS*03], [PLF*03], [PJ95], as well as work for in situ approaches [CMY*12]. In particular, this work focuses on in transit methods, where data being produced from the XGC1 [CKD*09] simulation in Singapore is moved asynchronously over the network to compute resources in Georgia, USA where the data will be visualized and queried. Their in transit method uses ADIOS [LLT*14], the middleware system that has a variety of different transport methods, including Dataspaces [DPK10], FlexPath [DBE*14], and ICEE [CWW*]. On the simulation side, 162MB of field and 62GB of particle data are produced every time step, which ADIOS then transfers to local staging nodes. From here, the field data is transferred over the WAN to Georgia where data consumers can visualize the data, steer the simulation, query the particle data, and track individual particles over time, all before the next time step begins (in 10 seconds). In terms of viewpoint, the user can interactively choose an area of interest which will determine the camera placement.

Weather.

Work by Ellsworth et al. [EGH*06] describe a time-critical visualization pipeline for weather forecasting using the fourth generation Goddard Earth Observing System (GEOS4) simulation code. The GEOS4 simulation is run under tight time constraints four times a day which requires the visualization to be performed with minimal overhead so they can be made available to forecasters at the National Hurricane Center. The visualization had to be performed on data consisting of 23 million cells with up to seven 3D and four 2D fields per cell. To be able to visualize the data quickly with high resolutions, Ellsworth et al. decoupled the data, transferring the data to staging nodes where they compressed the data using MPEG encoding. The resulting MPEG streams are then sent to the remote sites where the completed time steps are shown in a continuous animation loop. Camera placement is moot in this circumstance as the data is always presented on a 2D world map with an overhead viewpoint.

The work by Slawinska et al. [SCW*13] integrates ADIOS into the Maya computational astrophysics simulation to allow physicists to analyze and visualize their data in situ. By utilizing the staging capabilities of ADIOS, Slawinska et al. are able to apply in situ techniques to analyze, reduce, and visualize their data with little impact on the simulation. This work was primarily an adaptation of Maya's workflow, so there was no discussion of camera placement nor types of visualizations performed.

Molecular Dynamics.

Before data scientists employ in situ techniques, it's important to determine if in situ is a right fit for their use-case. If the analysis routines are compute heavy, it may be better to use in transit techniques and move data to secondary compute resources, or maybe write to disk. There also needs to be sufficient memory to perform the analysis, which may not be possible for memory intensive simulations. For large-scale simulations, Malakar et al. [MVM*15] propose a routine for optimal scheduling of in situ analysis that is based on resource configurations and application demands. Based on the time and memory requirements of the simulation and the analysis kernels, their program will recommend which analyses can be done in situ within the constraints, prioritize the analyses, while also taking into consideration the expected I/O costs. Overall the contributions of this work is four-fold [MVM*15]:

- Formulation of optimization problem for scheduling in situ analyses.
- Recommendation for performing in situ analyses based on their resource usage and available system resources.
- Performance modeling of in situ analysis routines.
- Demonstration of in situ analyses execution with the proposed optimization schedules with two exemplar applications, LAMMPS [Pli95] and FLASH [VHP*11], on a leadership supercomputing system.

The issue with this work is that after calculating which analyses to perform it also decides how many times to perform each analysis, then evenly spaces the analyses among the total number of time steps. Thus potentially missing important phenomena if the analyses are not performed at the correct time step.

Fluid Mechanics.

In order to visualize their data in situ, Lorendeau et al. [LFR13] integrate Catalyst [ABG*15] into Code_Saturn [AMS04], an open source Computational Fluid Dynamics (CFD) code designed to solve the Navier-Stokes equations for 2D, 2D axisymmetric, and 3D flows. With increasing computational power, researchers at the Electricité de France (EDF) were beginning to spend the majority of their time writing and reading their data. By integrating Catalyst the researchers were able to work around the growing I/O gap and were now able to visualize their data in situ with only 10% added overhead on runs with 204M hexahedral elements and 3,600 cores on Ivanoe, their corporate EDF supercomputer. Viewpoints and other parameters were determined based smaller test runs.

Also wanting to visualize their data in situ, Yi et al. [YRFB14] integrate Catalyst into the simulation Parallel Hierarchic Adaptive Stabilized Transient Analysis (PHASTA) [Jan99], an open source codebase used to solve compressible and incompressible Navier-Stokes equations. They tested their integration on both Oak Ridge National Lab's Titan and Argonne National Lab's Mira supercomputers. On Titan, using 18,432 cores and having 167M tetrahedral elements, the in situ visualization added a 10% execution overhead. Whereas on Mira, using 32,768 cores, the in situ visualization caused an additional 30% overhead. While they added steering capabilities and checkpoints so they could "rewind" and alter simulation parameters in situ, their initial conditions and viewpoints are based small test runs.

6. Evaluation

Over the course of this survey we implemented 14 of the 26 metrics to determine which are palatable for in situ implementation. Wanting to keep computational overheads low and memory footprints small, we immediately disqualified the following metrics for being unfit for in situ without significant work parallelizing the algorithms:$VQ_8, VQ_9, VQ_{17} - VQ_{26}$.

Of the remaining metrics, we applied them to several scientific datasets of varying size, measuring their average execution time as well as visually analyzing the images each metric determines to be the best and worst.

For this research we use isosurfaces of individual time steps from the following ECP datasets:

- **ExaSky-Nyx** a cosmological simulation: three different time steps with 19,280, 143,059, and 55,544 triangles.
- **ExaAM Truchas** a metallurgy casting simulation: three different time steps with 18,473, 6,474, and 21,255 triangles.
- **ExaConstit** a metallurgy simulation: two different time steps with 938,862, and 135,109 triangles.

Table 2 shows the average execution time for each metric per viewpoint for each time step. Some metrics can be computed during the rendering process, thus requiring little time to compute. But even those that require extra computations, none exceeded 2 seconds and most were significantly faster.

Figures 18-25 show the best and worst images for each metric for the eight time steps. Notice that there are viewpoints that are consistently favored among the majority of metrics, but overall the results could be wildly different depending on the data set.

In terms of findings, each metric has pros and cons. For the area metrics, they favor images that maximize pixel resolution, but in doing so can have lots of occlusions and little depth. For the silhouette metrics, they favor images with jagged silhouettes, which, in some cases, reduces occlusions. Similarly for depth, by maximizing the viewable depth of the image this prevents any large components from dominating.

In conclusion, we've realized is that no one metric will find the best image, but maybe a combination of metrics can. Additionally, the metrics implemented are purely based on the geometry of the data, what is needed is a new metric that also takes into account the field data. We believe an entropy calculation on the field data in conjunction with geometric metrics will result in a scientifically important image.

References

[ABG*15] Ayachit U., Bauer A., Geveci B., O'Leary P., Moreland K., Fabian N., Mauldin J.: 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* (New York, NY, USA, 2015), ISAV2015, ACM, pp. 25–29.

[AGL05] Ahrens J., Geveci B., Law C.: Paraview: An end-user tool for large data visualization. *Visualization Handbook* (01 2005).

[AJO*14] Ahrens J., Jourdain S., O'Leary P., Patchett J., Rogers D. H., Petersen M.: 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* (Piscataway, NJ, USA, 2014), SC '14, IEEE Press, pp. 424–434.

[AMS04] Archambeau F., Méchitoua N., Sakiz M.: Code saturne: A finite volume code for the computation of turbulent incompressible flows - industrial applications.

[Arn88] Arnheim R.: The power of center. University of California Press.

[AWW*16] Ayachit U., Whitlock B., Wolf M., Loring B., Geveci B., Lonie D., Bethel E. W.: The sensei generic in situ interface. In *Proceedings of the 2Nd Workshop on In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization* (Piscataway, NJ, USA, 2016), ISAV '16, IEEE Press, pp. 40–44.

[BAB*12] Bennett J. C., Abbasi H., Bremer P.-T., Grout R., Gyulassy A., Jin T., Klasky S., Kolla H., Parashar M., Pascucci V., Pebay P., Thompson D., Yu H., Zhang F., Chen J.: Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Los Alamitos, CA, USA, 2012), SC '12, IEEE Computer Society Press, pp. 49:1–49:9.

[BDP99] Barral P., Dorme G., Plemenos D.: Visual understanding of a scene by automatic movement of a camera. In *GraphicCon '99* (January 1999).

[BDP00] Barral P., Dorme G., Plemenos D.: Scene understanding techniques using a virtual camera. In *Eurographics 2000 - Short Presentations* (2000), Eurographics Association.

[Bet00a] Bethel E. W.: Visapult: A prototype remote and distributed visualization application and framework.

[Bet00b] Bethel W.: Visualization dot com. *IEEE Computer Graphics and Applications 20*, 3 (May 2000), 17–20.

[BFS11] Bonaventura X., Feixas M., Sbert M.: Viewpoint information. *21st International Conference on Computer Graphics and Vision, GraphiCon'2011 - Conference Proceedings* (01 2011).

[BFS*18] Bonaventura X., Feixas M., Sbert M., Chuang L., Wallraven C.: A survey of viewpoint selection methods for polygonal models. *Entropy 20*, 5 (2018).

[Bir33] Birkhoff G.: Aesthetic measure. *Harvard University Press, Cambridge, Massachusetts* (1933).

[Bir56] Birkhoff G.: Mathematics of aesthetics. *J.R. Newman (ed.) The World of Mathematics 4* (1956).

[Bla87a] Blahut R. E.: *Principles and Practice of Information Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[Bla87b] Blahut R. E.: *Principles and Practice of Information Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[BPS97] Bajaj C. L., Pascucci V., Schikore D. R.: The contour spectrum. In *Proceedings of the 8th Conference on Visualization '97* (Los Alamitos, CA, USA, 1997), VIS '97, IEEE Computer Society Press, pp. 167–ff.

[BR82] Burbea J., Rao C.: On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory 28*, 3 (May 1982), 489–495.

[BS05] Bordoloi U., Shen H.: View selection for volume rendering. In *16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, October 23-28, 2005* (2005), pp. 487–494.

[BSS*03] Bethel W., Siegerist C., Shalf J., Shetty P., Jankun-Kelly T., Kreylos O., Ma K.-L.: Visportal: Deploying grid-enabled visualization tools through a web-portal interface.

[BTB99] Blanz V., Tarr M. J., Bülthoff H. H.: What object attributes determine canonical views? *Perception 28 5* (1999), 575–99.

[BTS*18] Bujack R., Turton T. L., Samsel F., Ware C., Rogers D. H., Ahrens J.: The good, the bad, and the ugly: A theoretical framework for the assessment of continuous colormaps. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 923–933.

[But03] Butts D. A.: How much information is associated with a particular stimulus? *Network: Computation in Neural Systems 14*, 2 (2003), 177–187. PMID: 12790180.

[CBW*12] Childs H., Brugger E., Whitlock B., Meredith J., Ahern S., Pugmire D., Biagas K., Miller M., Harrison C., Weber G. H., Krishnan H., Fogal T., Sanderson A., Garth C., Bethel E. W., Camp D., Rübel O., Durant M., Favre J. M., Navrátil P.: VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. Oct 2012, pp. 357–372.

[CKD*09] Chang C. S., Ku S., Diamond P. H., Lin Z., Parker S., Hahm T. S., Samatova N.: Compressed ion temperature gradient turbulence in diverted tokamak edge. *Physics of Plasmas 16*, 5 (2009), 056108.

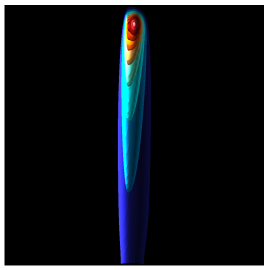
[CLR*19] Childs H., Loring B., Rogers D., Larsen M., Harrison C., Rizzi S., Whitlock B., Thompson D.: In situ and visualization tutorial with sensei and ascent. In *International Conference for High Performance Computing, Networking, Storage and Analysis* (2019), SC '19.

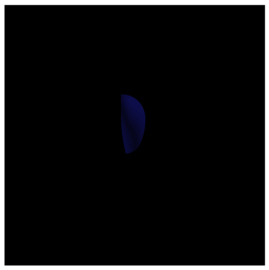
[CMY*12] Childs H., Ma K.-L., Yu H., Whitlock B., Meredith J., Favre J., Klasky S., Podhorszki N., Schwan K., Wolf M., et al.: *In situ processing*. Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2012.

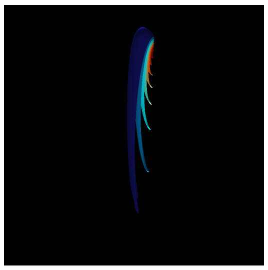
[CT06] Cover T. M., Thomas J. A.: *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

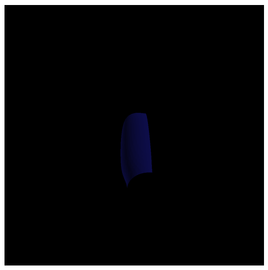
[CWW*] Choi J. Y., Wu K., Wu J. C., Sim A., Liu Q. G., Wolf M., Chang C., Klasky S.: Icee: Wide-area in transit data processing framework for near real-time scientific applications.

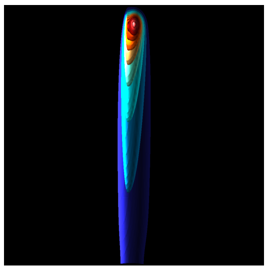
[DBE*14] Dayal J., Bratcher D., Eisenhauer G., Schwan K., Wolf M., Zhang X., Abbasi H., Klasky S., Podhorszki N.: Flexpath: Type-based publish/subscribe system for large-scale science analytics. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (May 2014), pp. 246–255.

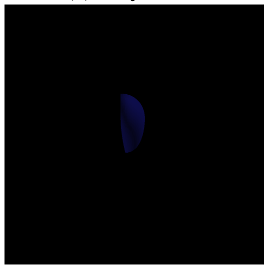

(a) VQ_1 Best

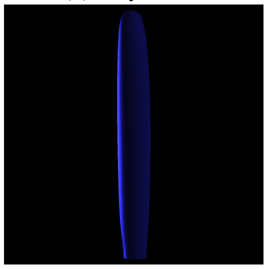

(b) VQ_1 Worst

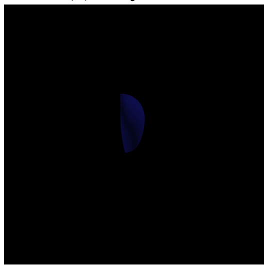

(c) VQ_2 Best

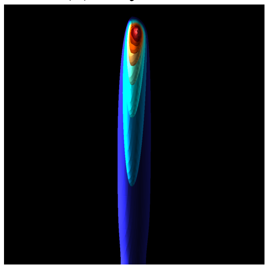

(d) VQ_2 Worst

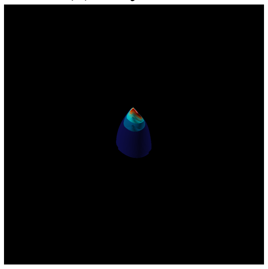

(e) VQ_3 Best

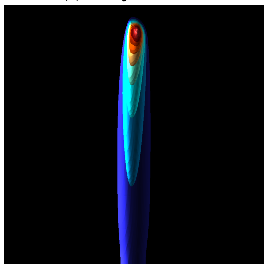

(f) VQ_3 Worst

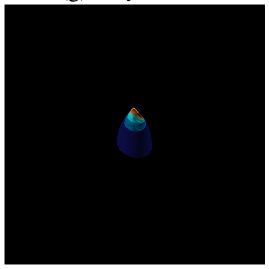

(g) VQ_4 Best

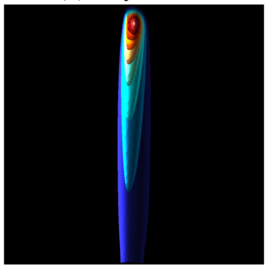

(h) VQ_4 Worst

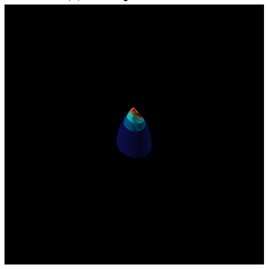

(i) VQ_5 Best

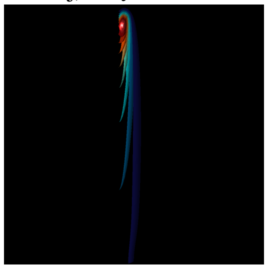

(j) VQ_5 Worst

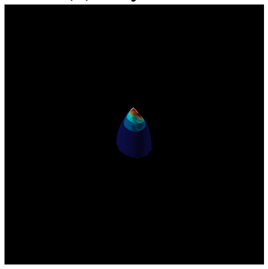

(k) VQ_6 Best

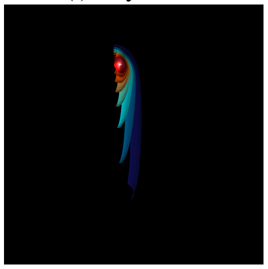

(l) VQ_6 Worst

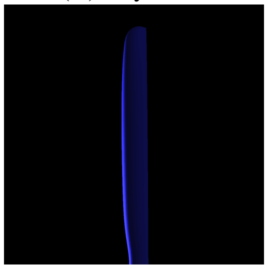

(m) VQ_7 Best

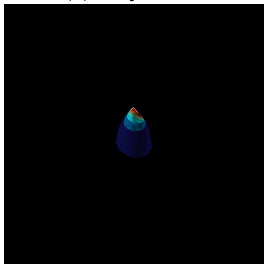

(n) VQ_7 Worst

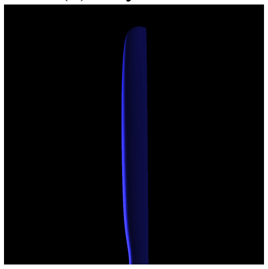

(o) VQ_{10} Best

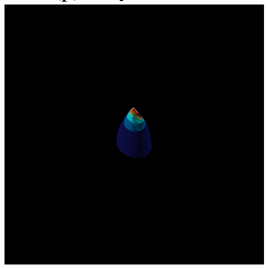

(p) VQ_{10} Worst

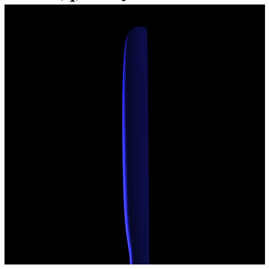

(q) VQ_{11} Best

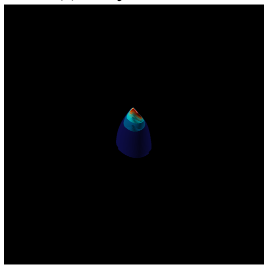

(r) VQ_{11} Worst

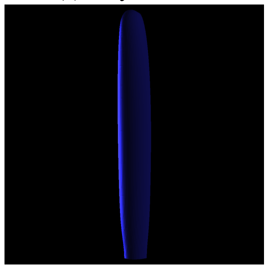

(s) VQ_{12} Best

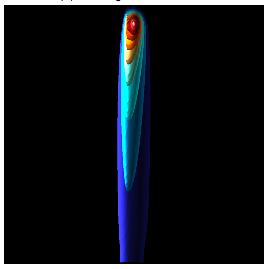

(t) VQ_{12} Worst

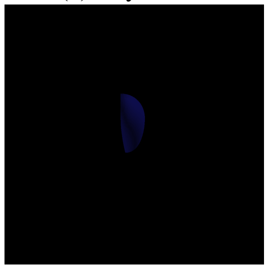

(u) VQ_{13} Best

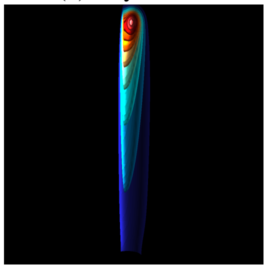

(v) VQ_{13} Worst


(w) VQ_{14} Best


(x) VQ_{14} Worst


(y) VQ_{15} Best


(z) VQ_{15} Worst


(aa) VQ_{16} Best

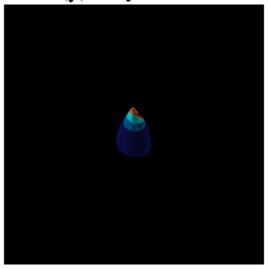
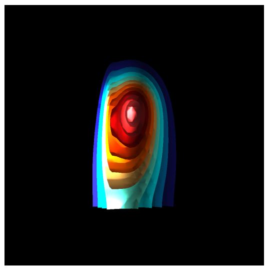
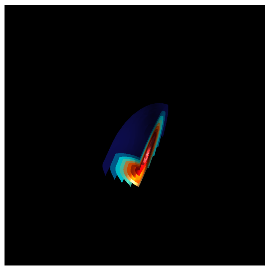
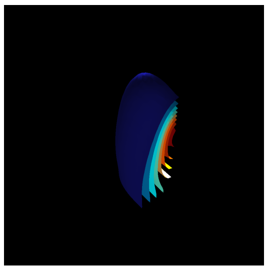
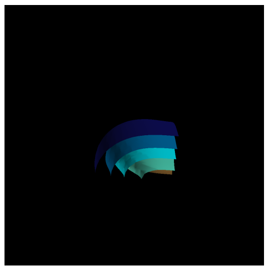

(ab) VQ_{16} Worst

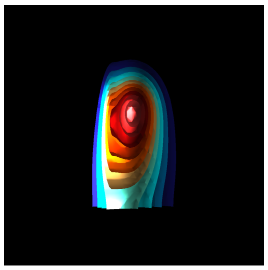
Figure 18: The best and worst images for the implemented metrics on the ExaAM time step with 18,473 triangles.

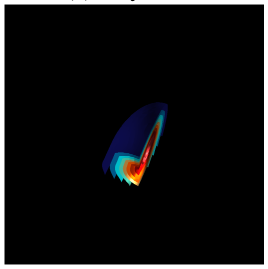

(a) VQ_1 Best

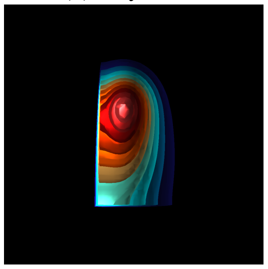

(b) VQ_1 Worst

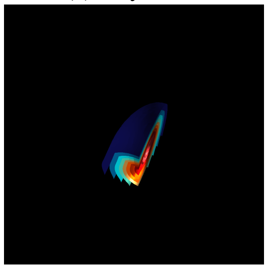

(c) VQ_2 Best

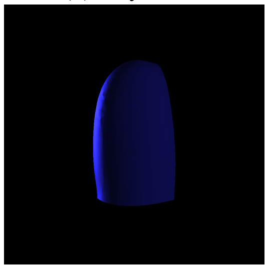

(d) VQ_2 Worst

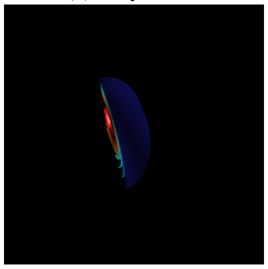

(e) VQ_3 Best

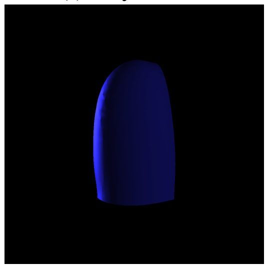

(f) VQ_3 Worst

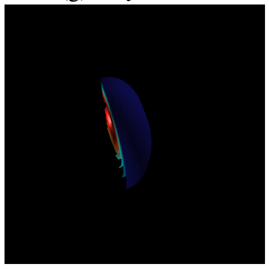

(g) VQ_4 Best

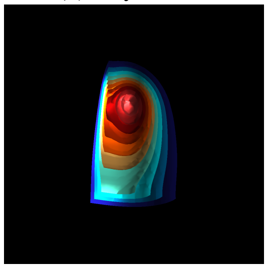

(h) VQ_4 Worst

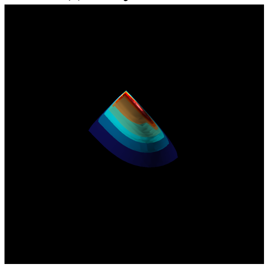

(i) VQ_5 Best

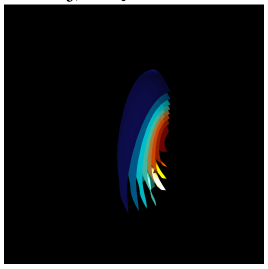

(j) VQ_5 Worst

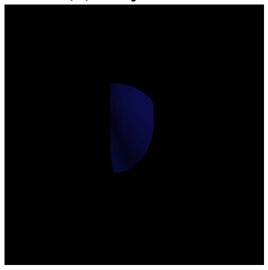

(k) VQ_6 Best

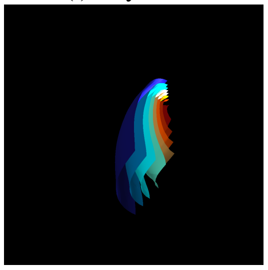

(l) VQ_6 Worst

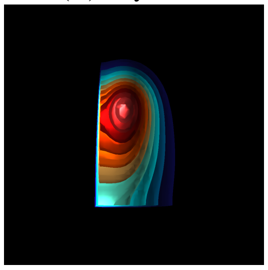

(m) VQ_7 Best

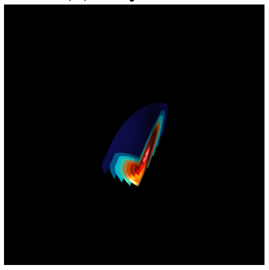

(n) VQ_7 Worst

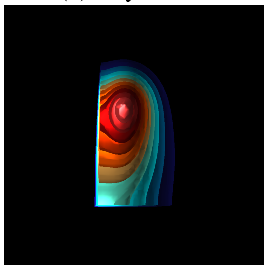

(o) VQ_{10} Best

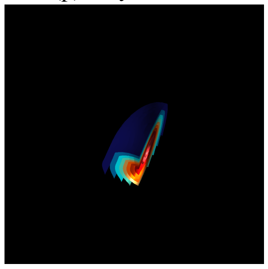

(p) VQ_{10} Worst

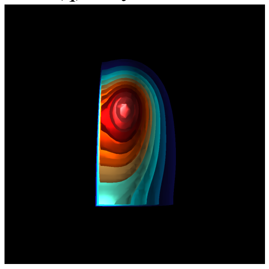

(q) VQ_{11} Best

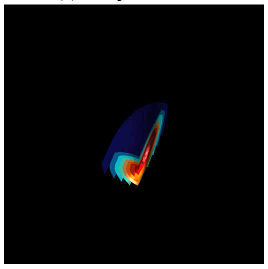

(r) VQ_{11} Worst

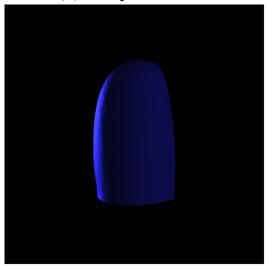

(s) VQ_{12} Best

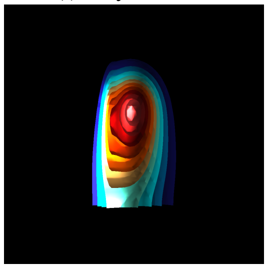

(t) VQ_{12} Worst

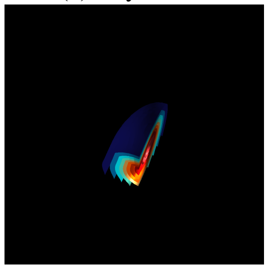

(u) VQ_{13} Best

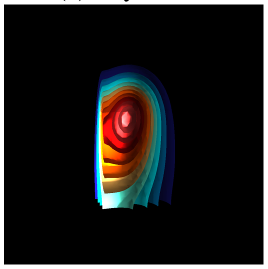

(v) VQ_{13} Worst


(w) VQ_{14} Best


(x) VQ_{14} Worst


(y) VQ_{15} Best


(z) VQ_{15} Worst


(aa) VQ_{16} Best

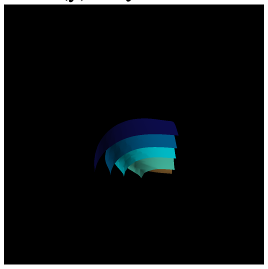
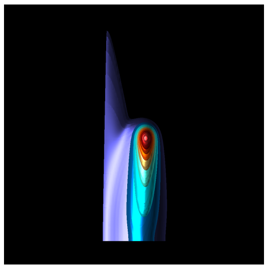
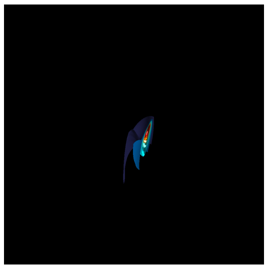
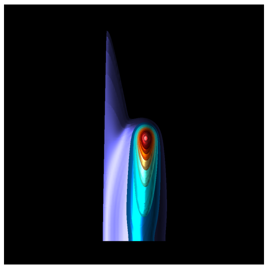
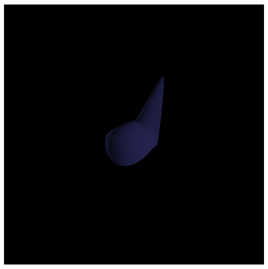

(ab) VQ_{16} Worst

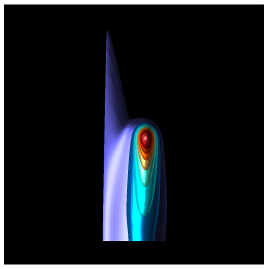
Figure 19: The best and worst images for the implemented metrics on the ExaAM time step with 6,474 triangles.

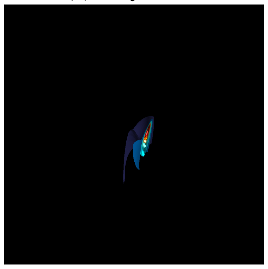

(a) VQ_1 Best

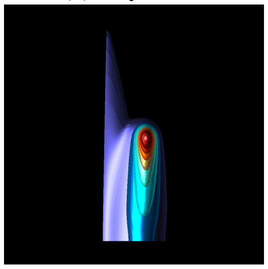

(b) VQ_1 Worst

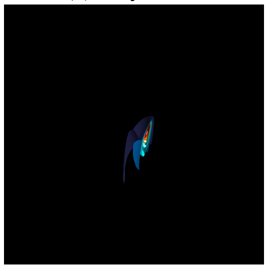

(c) VQ_2 Best

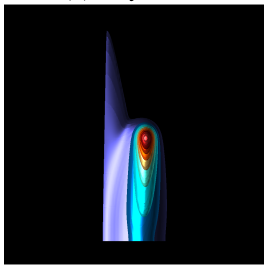

(d) VQ_2 Worst

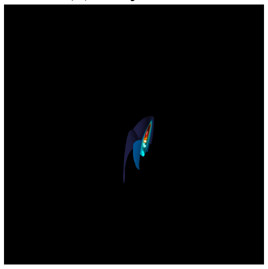

(e) VQ_3 Best

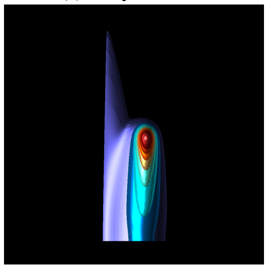

(f) VQ_3 Worst

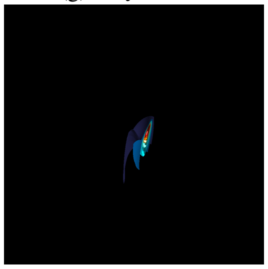

(g) VQ_4 Best

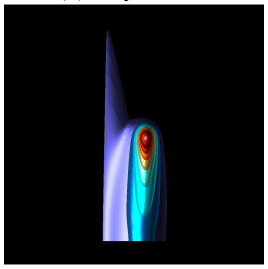

(h) VQ_4 Worst

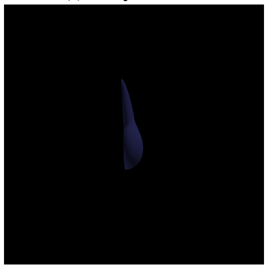

(i) VQ_5 Best

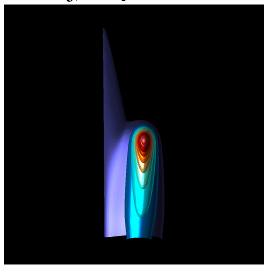

(j) VQ_5 Worst

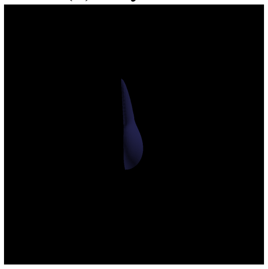

(k) VQ_6 Best

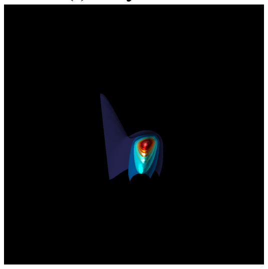

(l) VQ_6 Worst

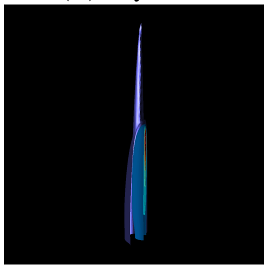

(m) VQ_7 Best

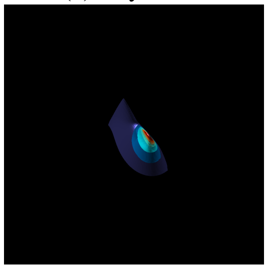

(n) VQ_7 Worst

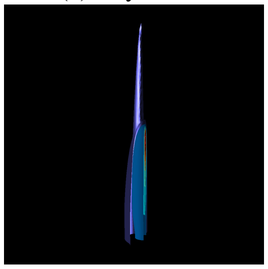

(o) VQ_{10} Best

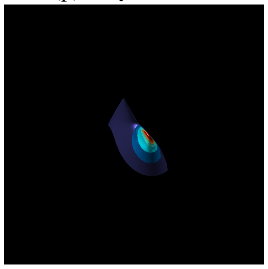

(p) VQ_{10} Worst

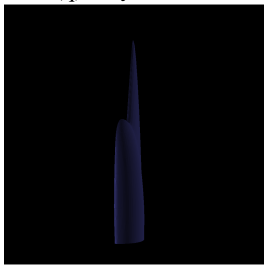

(q) VQ_{11} Best

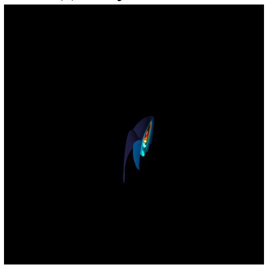

(r) VQ_{11} Worst

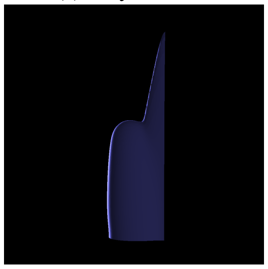

(s) VQ_{12} Best

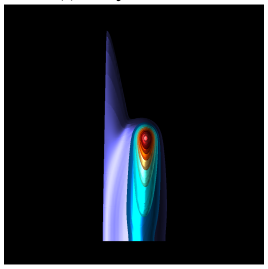

(t) VQ_{12} Worst

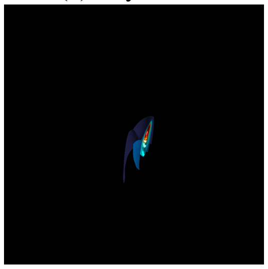

(u) VQ_{13} Best

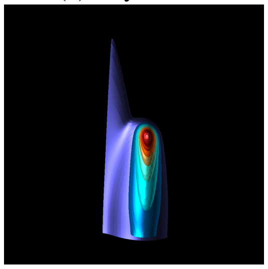

(v) VQ_{13} Worst


(w) VQ_{14} Best


(x) VQ_{14} Worst


(y) VQ_{15} Best


(z) VQ_{15} Worst


(aa) VQ_{16} Best

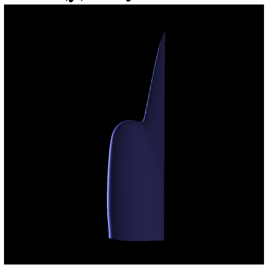

(ab) VQ_{16} Worst

Figure 20: The best and worst images for the implemented metrics on the ExaAM time step with 21,255 triangles.

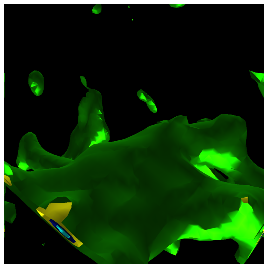
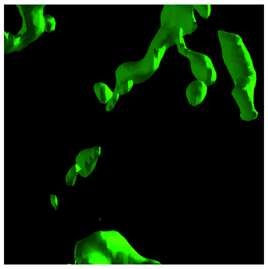
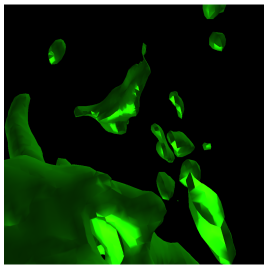
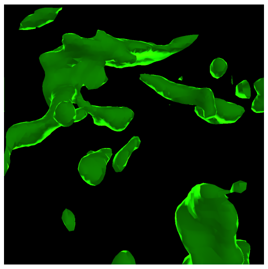
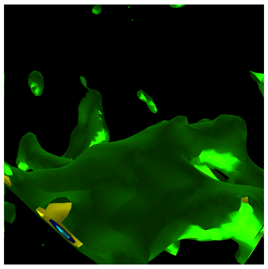
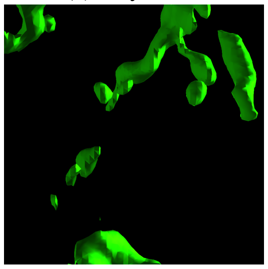
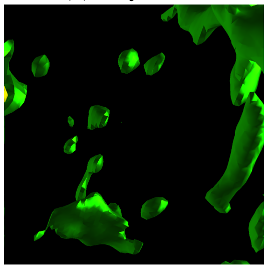
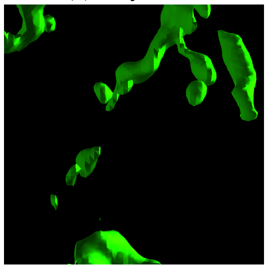
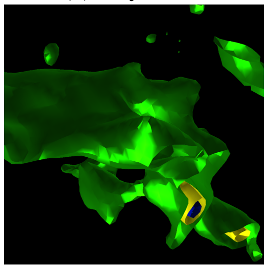
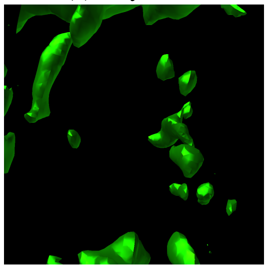
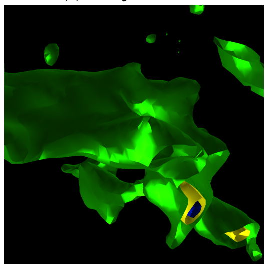
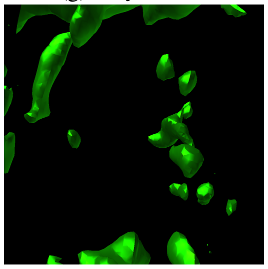
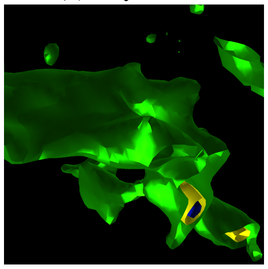
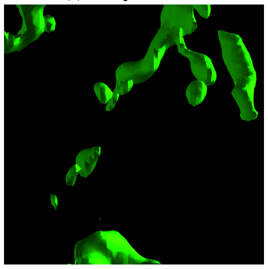
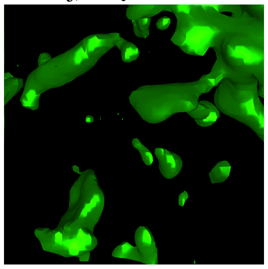
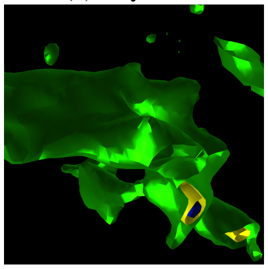
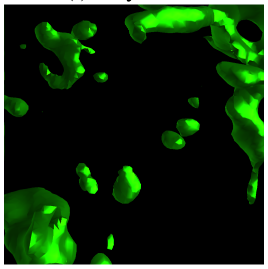
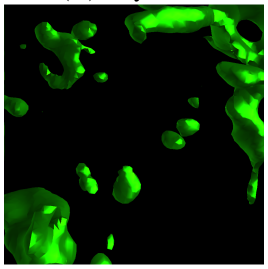
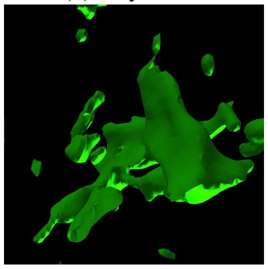
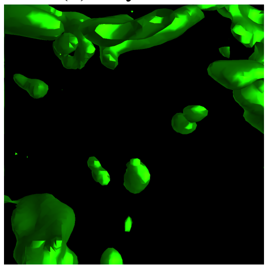
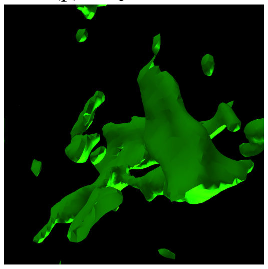
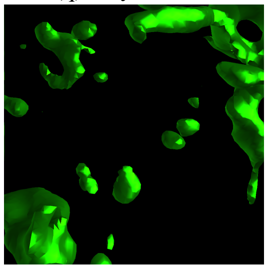
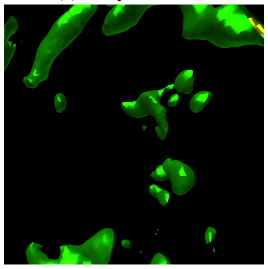
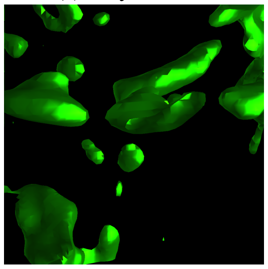
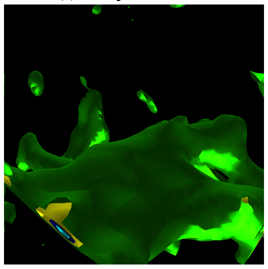
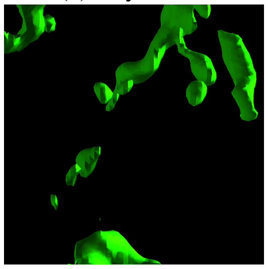
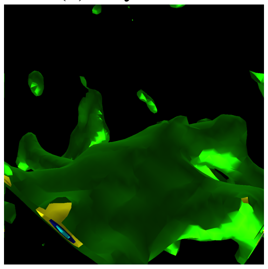
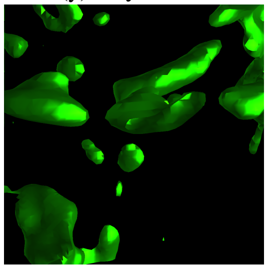
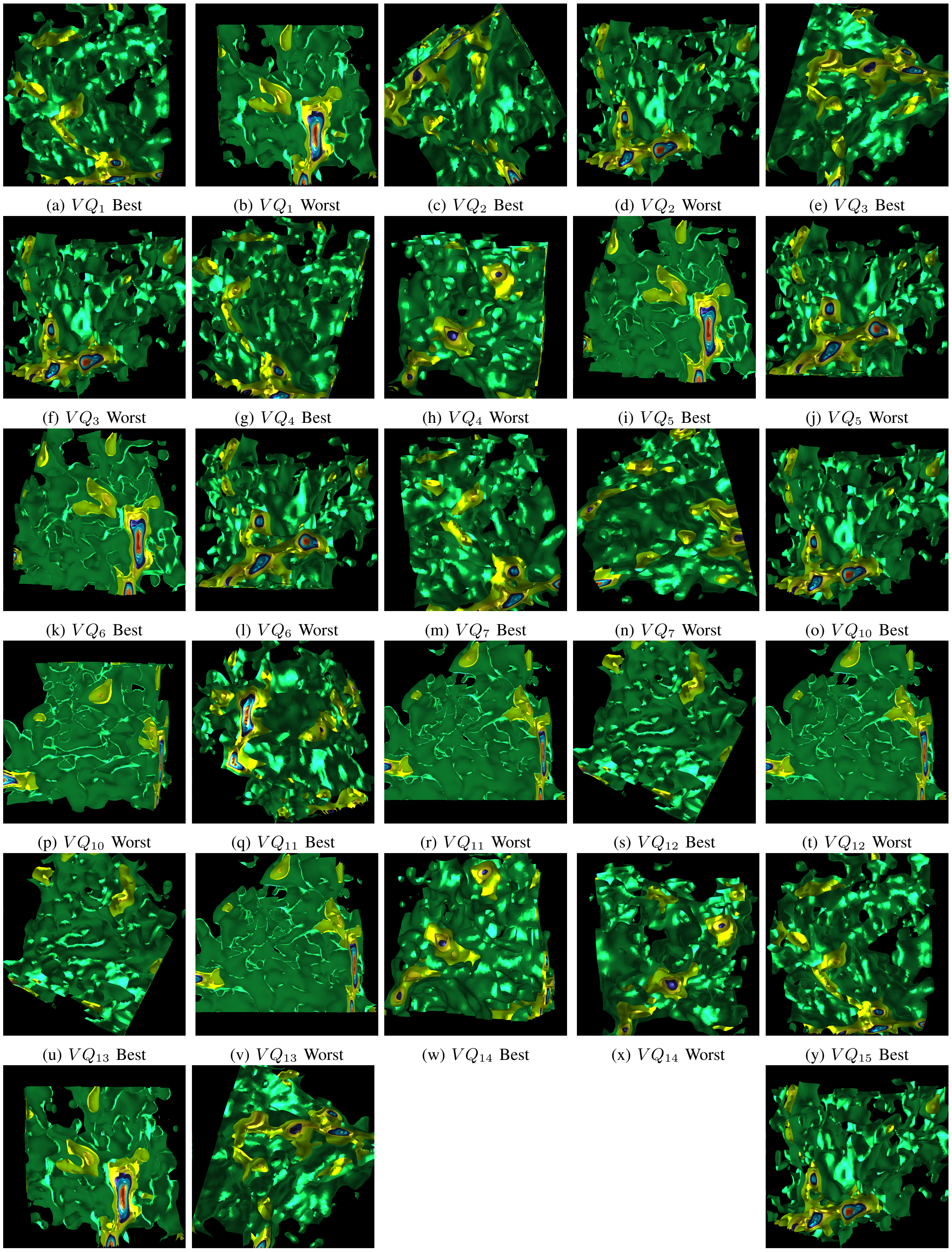

(a) VQ_1 Best

(b) VQ_1 Worst

(c) VQ_2 Best

(d) VQ_2 Worst

(e) VQ_3 Best

(f) VQ_3 Worst

(g) VQ_4 Best

(h) VQ_4 Worst

(i) VQ_5 Best

(j) VQ_5 Worst

(k) VQ_6 Best

(l) VQ_6 Worst

(m) VQ_7 Best

(n) VQ_7 Worst

(o) VQ_{10} Best

(p) VQ_{10} Worst

(q) VQ_{11} Best

(r) VQ_{11} Worst

(s) VQ_{12} Best

(t) VQ_{12} Worst

(u) VQ_{13} Best

(v) VQ_{13} Worst

(w) VQ_{14} Best

(x) VQ_{14} Worst

(y) VQ_{15} Best

(z) VQ_{15} Worst

(aa) VQ_{16} Best

(ab) VQ_{16} Worst

Figure 21: The best and worst images for the implemented metrics on the ExaAM time step with 19,280 triangles.



(a) VQ_1 Best (b) VQ_1 Worst (c) VQ_2 Best (d) VQ_2 Worst (e) VQ_3 Best

(f) VQ_3 Worst (g) VQ_4 Best (h) VQ_4 Worst (i) VQ_5 Best (j) VQ_5 Worst

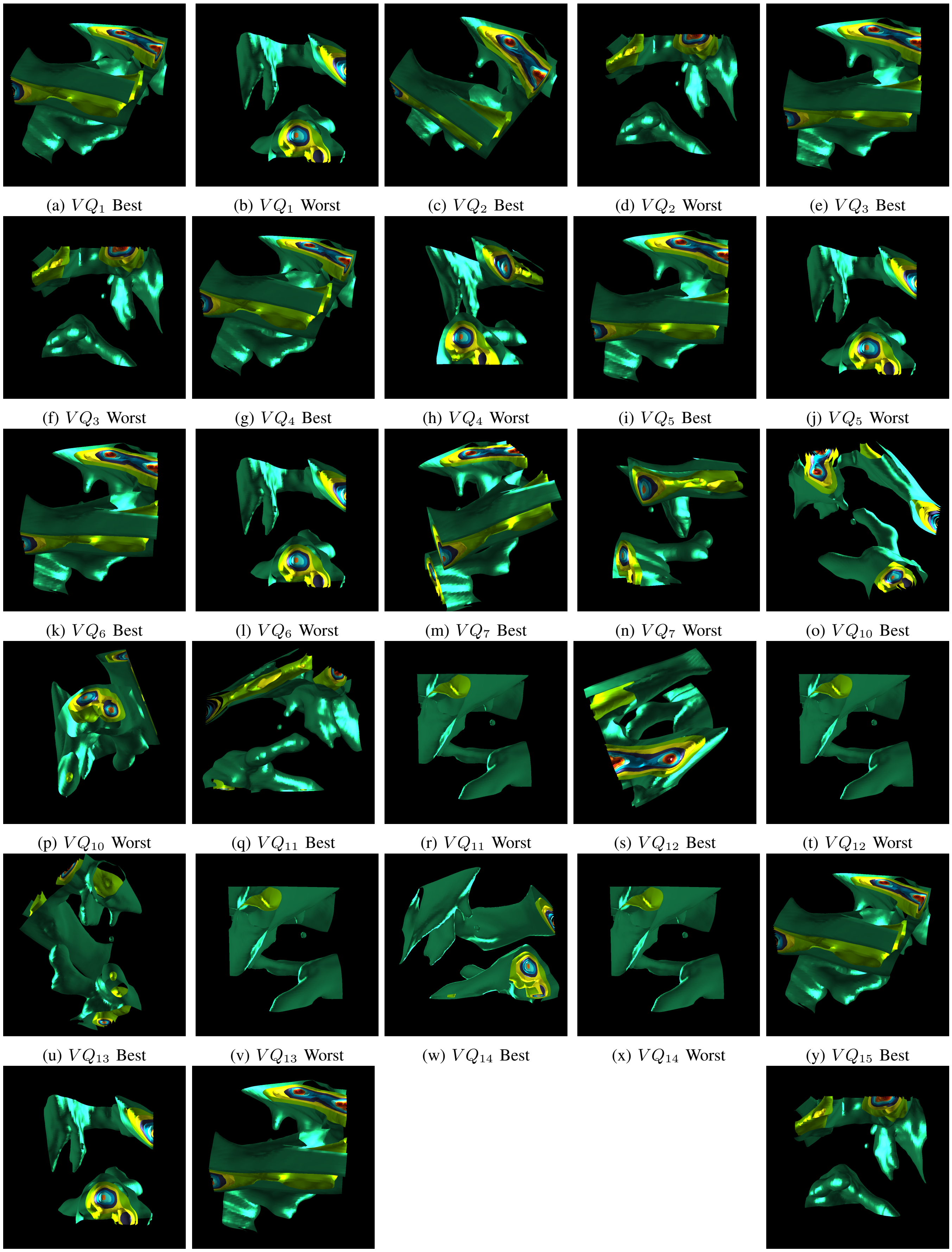
(k) VQ_6 Best (l) VQ_6 Worst (m) VQ_7 Best (n) VQ_7 Worst (o) VQ_{10} Best

(p) VQ_{10} Worst (q) VQ_{11} Best (r) VQ_{11} Worst (s) VQ_{12} Best (t) VQ_{12} Worst

(u) VQ_{13} Best (v) VQ_{13} Worst (w) VQ_{14} Best (x) VQ_{14} Worst (y) VQ_{15} Best

(z) VQ_{15} Worst (aa) VQ_{16} Best (ab) VQ_{16} Worst

Figure 22: The best and worst images for the implemented metrics on the ExaAM time step with 143,059 triangles.



(a) VQ_1 Best (b) VQ_1 Worst (c) VQ_2 Best (d) VQ_2 Worst (e) VQ_3 Best

(f) VQ_3 Worst (g) VQ_4 Best (h) VQ_4 Worst (i) VQ_5 Best (j) VQ_5 Worst

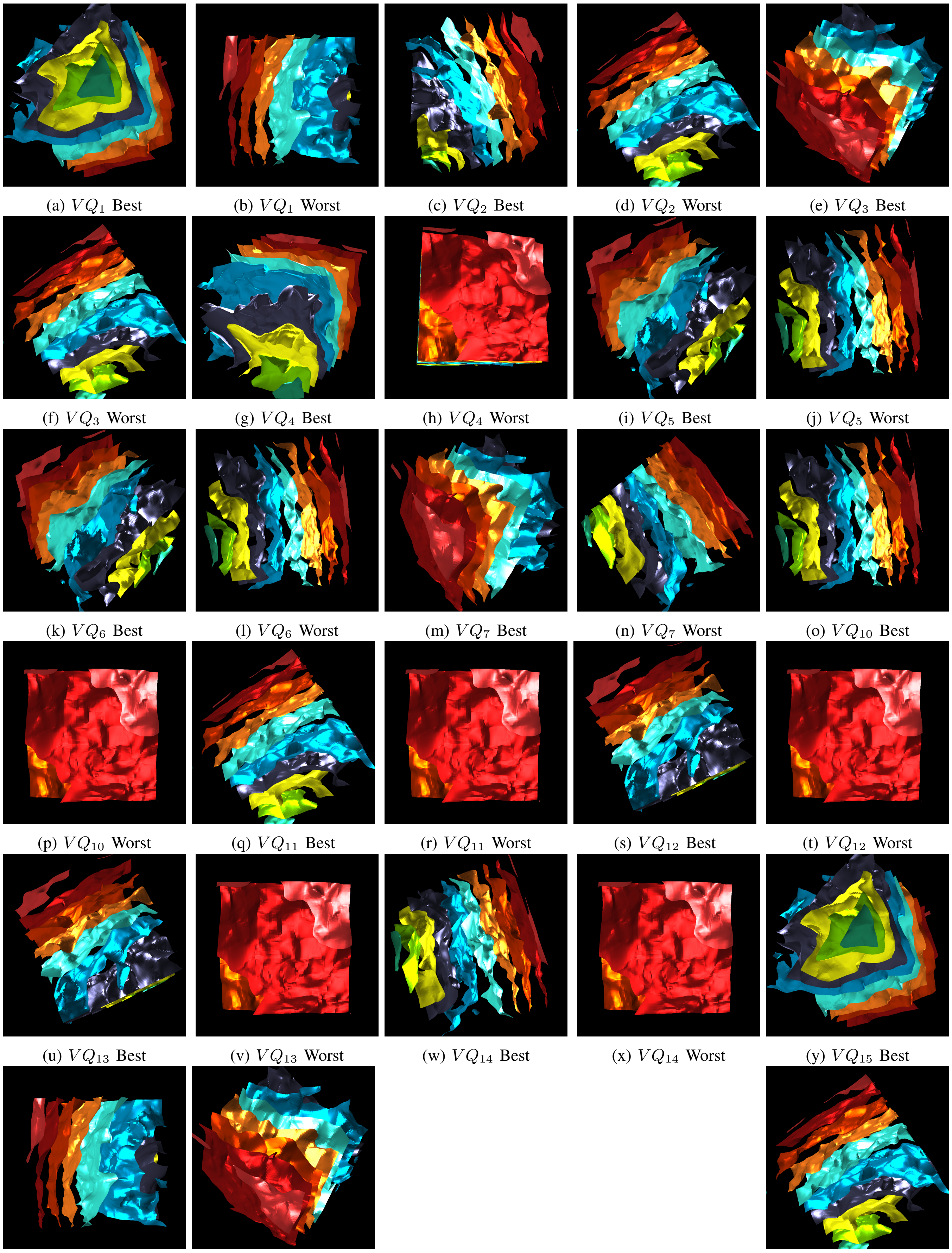
(k) VQ_6 Best (l) VQ_6 Worst (m) VQ_7 Best (n) VQ_7 Worst (o) VQ_{10} Best

(p) VQ_{10} Worst (q) VQ_{11} Best (r) VQ_{11} Worst (s) VQ_{12} Best (t) VQ_{12} Worst

(u) VQ_{13} Best (v) VQ_{13} Worst (w) VQ_{14} Best (x) VQ_{14} Worst (y) VQ_{15} Best

(z) VQ_{15} Worst (aa) VQ_{16} Best (ab) VQ_{16} Worst

Figure 23: The best and worst images for the implemented metrics on the ExaAM time step with 55,544 triangles.



(a) VQ_1 Best (b) VQ_1 Worst (c) VQ_2 Best (d) VQ_2 Worst (e) VQ_3 Best

(f) VQ_3 Worst (g) VQ_4 Best (h) VQ_4 Worst (i) VQ_5 Best (j) VQ_5 Worst

(k) VQ_6 Best (l) VQ_6 Worst (m) VQ_7 Best (n) VQ_7 Worst (o) VQ_{10} Best

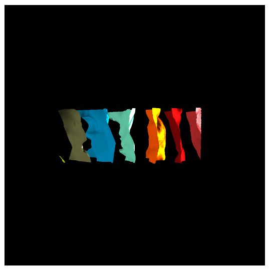
(p) VQ_{10} Worst (q) VQ_{11} Best (r) VQ_{11} Worst (s) VQ_{12} Best (t) VQ_{12} Worst

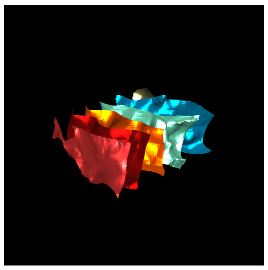
(u) VQ_{13} Best (v) VQ_{13} Worst (w) VQ_{14} Best (x) VQ_{14} Worst (y) VQ_{15} Best

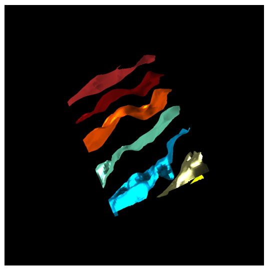
(z) VQ_{15} Worst (aa) VQ_{16} Best (ab) VQ_{16} Worst

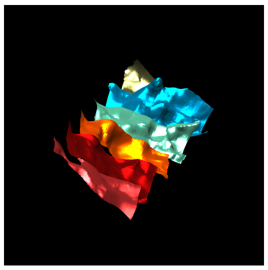
Figure 24: The best and worst images for the implemented metrics on the ExaAM time step with 938,862 triangles.

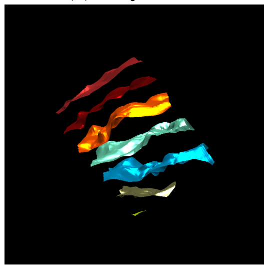
(a) VQ_1 Best

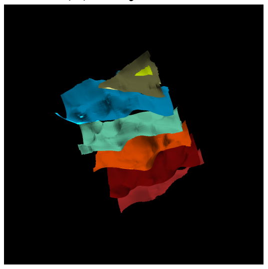

(b) VQ_1 Worst

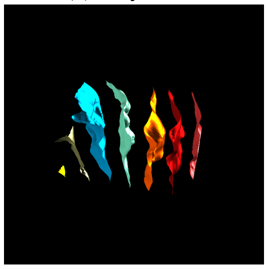

(c) VQ_2 Best

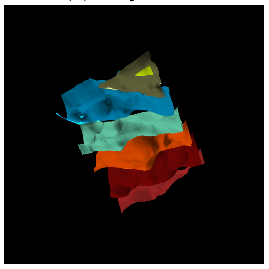

(d) VQ_2 Worst

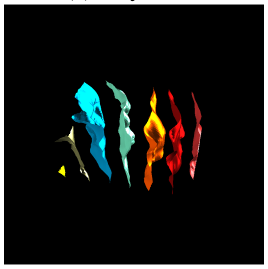

(e) VQ_3 Best

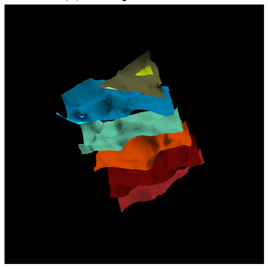

(f) VQ_3 Worst

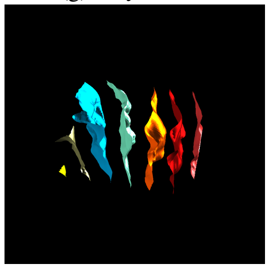

(g) VQ_4 Best

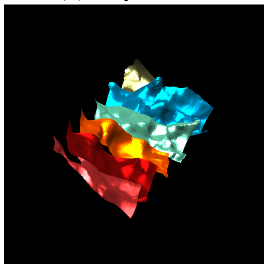

(h) VQ_4 Worst

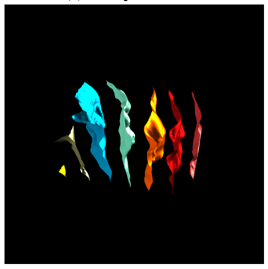

(i) VQ_5 Best

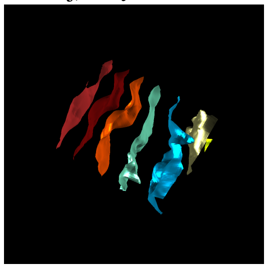

(j) VQ_5 Worst


(k) VQ_6 Best

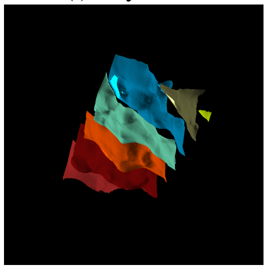

(l) VQ_6 Worst


(m) VQ_7 Best

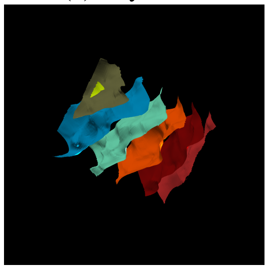

(n) VQ_7 Worst


(o) VQ_{10} Best

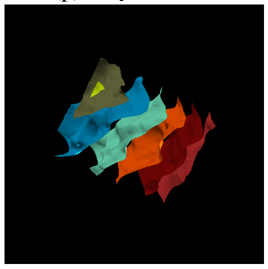
(p) VQ_{10} Worst


(q) VQ_{11} Best

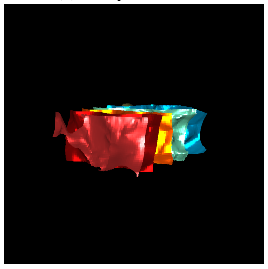
(r) VQ_{11} Worst

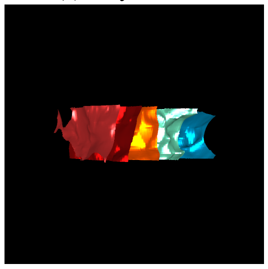

(s) VQ_{12} Best

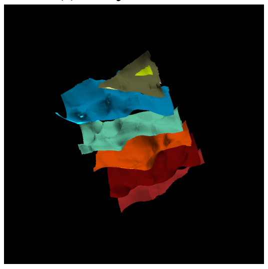
(t) VQ_{12} Worst

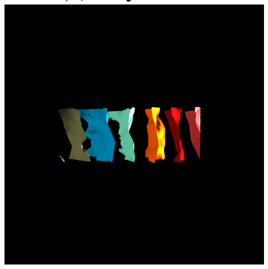

(u) VQ_{13} Best

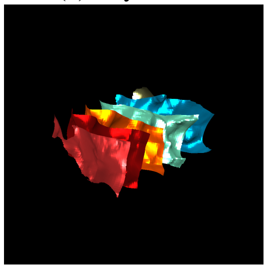
(v) VQ_{13} Worst


(w) VQ_{14} Best


(x) VQ_{14} Worst


(y) VQ_{15} Best


(z) VQ_{15} Worst


(aa) VQ_{16} Best

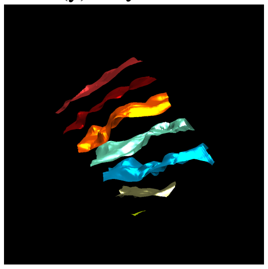

(ab) VQ_{16} Worst

Figure 25: The best and worst images for the implemented metrics on the ExaAM time step with 135,109 triangles.

Metric	ExaConstit 1 (938,862 △s) Time in μs	ExaConstit 2 (135,109 △s)	ExaSky 1 (55,544 △s)	ExaSky 2 (143,059 △s)	ExaSky 3 (19,280 △s)	ExaAM 1 (21,255 △s)	ExaAM 2 (6,474 △s)	ExaAM 3 (18,473 △s)
VQ_1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
VQ_2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
VQ_3	248065 μs	14830.5 μs	5311 μs	14185.3 μs	1929.03	2508.63 μs	305.72 μs	952.74 μs
VQ_4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
VQ_5	1.33009e+06 μs	268443 μs	626197 μs	1.36365e+06 μs	545438 μs	148650 μs	207043 μs	135033 μs
VQ_6	1.33009e+06 μs	268443 μs	143059 μs	1.36365e+06 μs	545438 μs	148650 μs	207043 μs	135033 μs
VQ_7	1.25724e+06 μs	33725.3 μs	615655 μs	1.33423e+06 μs	531735 μs	143863 μs	202881 μs	133310 μs
VQ_{10}	17144.4 μs	12047.8 μs	14965.4 μs	20626.9 μs	15637.9 μs	11319.3 μs	11141.9 μs	12442.1 μs
VQ_{11}	17144.4 μs	12047.8 μs	14965.4 μs	20626.9 μs	15637.9 μs	11319.3 μs	11141.9 μs	12442.1 μs
VQ_{12}	17144.4 μs	12047.8 μs	14965.4 μs	20626.9 μs	15637.9 μs	11319.3 μs	11141.9 μs	12442.1 μs
VQ_{13}	17144.4 μs	12047.8 μs	14965.4 μs	20626.9 μs	15637.9 μs	11319.3 μs	11141.9 μs	12442.1 μs
VQ_{14}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
VQ_{15}	40476.5 μs	20735.9 μs	29617.2 μs	50017.6 μs	27389 μs	17836.3 μs	19834.8 μs	17445.5 μs
VQ_{16}	36.96 μs	37.52 μs	35.66 μs	45.04 μs	39.31 μs	36.85 μs	36.4 μs	36.1 μs

TABLE 2: The average execution time in microseconds for each metric per viewpoint for each of the eight datasets. N/A means the metric calculation is done in conjunction with the rasterization process.

[DLANL14] DSS Los Alamos National Lab .: Cinema tutorial. In *International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), SC '14.

[DM99] DeWeese M., Meister M. L. R.: How to measure the information gained from one symbol. *Network Computing Neural Systems 10* (1999), 325–40.

[DPK10] Docan C., Parashar M., Klasky S.: Dataspaces: an interaction and coordination framework for coupled simulation workflows. In *HPDC* (2010).

[EGH*06] Ellsworth D., Green B., Henze C., Moran P., Sandstrom T.: Concurrent visualization in a production supercomputing environment. *IEEE transactions on visualization and computer graphics 12* (09 2006), 997–1004.

[FS05] Feldman J., Singh M.: Information along contours and object boundaries. *Psychological Review 112*, 1 (1 2005), 243–252.

[FSG09] Feixas M., Sbert M., González F.: A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Trans. Appl. Percept. 6*, 1 (Feb. 2009), 1:1–1:23.

[GRMS01] Gooch B., Reinhard E., Moulding C., Shirley P.: Artistic composition for image creation. In *Rendering Techniques 2001* (Vienna, 2001), Gortler S. J., Myszkowski K., (Eds.), Springer Vienna, pp. 83–88.

[Gum02] Gumhold S.: Maximum entropy light source placement. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 275–282.

[GZ05] Garland M., Zhou Y.: Quadric-based simplification in any dimension. *ACM Trans. Graph. 24*, 2 (Apr. 2005), 209–239.

[HLB16] Harrison C., Larsen M., Brugger E.: A lightweight in situ visualization and analysis infrastructure for multi-physics hpc simulation codes, version 00, 12 2016.

[HM03] Halle M., Meng J. C.: Lightkit: a lighting system for effective visualization. *IEEE Visualization, 2003. VIS 2003.* (2003), 363–370.

[IKN98] Itti L., Koch C., Niebur E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (Nov 1998), 1254–1259.

[Jan99] Jansen K. E.: A stabilized finite element method for computing turbulence. *Computer Methods in Applied Mechanics and Engineering 174*, 3 (1999), 299 – 317.

[JPP02] Jolivet V., Plemenos D., Poulingeas P.: Inverse direct lighting with a monte carlo method and declarative modelling. In *International Conference on Computational Science* (2002).

[KCK*16] Kress J., Churchill R. M., Klasky S., Kim M., Childs H., Pugmire D.: Preparing for in situ processing on upcoming leading-edge supercomputers. *Supercomputing frontiers and innovations 3*, 4 (2016), 49–65.

[KK88] Kamada T., Kawai S.: A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, and Image Processing 41*, 1 (1988), 43 – 56.

[LAA*17] Larsen M., Ahrens J., Ayachit U., Brugger E., Childs H., Geveci B., Harrison C.: The alpine in situ infrastructure: Ascending from the ashes of strawman. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization* (New York, NY, USA, 2017), ISAV'17, ACM, pp. 42–46.

[LBC*15] Larsen M., Brugger E., Childs H., Eliot J., Griffin K., Harrison C.: Strawman: A batch in situ visualization and analysis infrastructure for multi-physics simulation codes. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (New York, NY, USA, 2015), ISAV2015, ACM, pp. 30–35.

[LBMC16] Lessley B., Binyahib R., Maynard R., Childs H.: External facelist calculation with data-parallel primitives. In *EGPGV* (2016), pp. 11–20.

[LFR13] Lorendeau B., Fournier Y., Ribes A.: In-situ visualization in fluid mechanics using catalyst: A case study for code saturne. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)* (Oct 2013), pp. 53–57.

[LHK*16] Larsen M., Harrison C., Kress J., Pugmire D., Meredith J. S., Childs H.: Performance modeling of in situ rendering. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2016), IEEE, pp. 276–287.

[Lin91] Lin J.: Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory 37*, 1 (Jan 1991), 145–151.

[LLCC17] Li S., Larsen M., Clyne J., Childs H.: Performance impacts of in situ wavelet compression on scientific simulations. In *Proceedings of the in situ infrastructures on enabling extreme-scale analysis and visualization* (2017), ACM, pp. 37–41.

[LLT*14] LIU Q., LOGAN J., TIAN Y., ABBASI H., PODHORSZKI N., CHOI J. Y., KLASKY S., TCHOUA R., LOFSTEAD J., OLDFIELD R., PARASHAR M., SAMATOVA N., SCHWAN K., SHOSHANI A., WOLF M., WU K., YU W.: Hello adios: The challenges and lessons of developing leadership class i/o frameworks. *Concurr. Comput. : Pract. Exper. 26*, 7 (May 2014), 1453–1473.

[LMC*17] LI S., MARSAGLIA N., CHEN V., SEWELL C. M., CLYNE J. P., CHILDS H.: Achieving portable performance for wavelet compression using data parallel primitives. In *EGPGV* (2017), pp. 73–81.

[LS07] LI L., SHEN H.: Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (May 2007), 630–640.

[LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 659–666.

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 389–400.

[MCHM10] MARCHESIN S., CHEN C., HO C., MA K.: View-dependent streamlines for 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov 2010), 1578–1586.

[Mes17] MESSINA P.: *The Exascale Computing Project*. Computing in Science and Engineering 19, 3, 2017.

[MSU*16] MORELAND K., SEWELL C., USHER W., LO L., MEREDITH J., PUGMIRE D., KRESS J., SCHROOTS H., MA K., CHILDS H., LARSEN M., CHEN C., MAYNARD R., GEVECI B.: Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications 36*, 3 (May 2016), 48–58.

[MVM*15] MALAKAR P., VISHWANATH V., MUNSON T., KNIGHT C., HERELD M., LEYFFER S., PAPKA M. E.: Optimal scheduling of in-situ analysis for large-scale scientific simulations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2015), SC '15, ACM, pp. 52:1–52:11.

[PB96] PLEMENOS D., BENAYADA M.: Intelligent display in scene modelling. new techniques to automatically compute good views. In *GraphiCon'96* (Saint Petersburg (Russia), July 1996).

[PF92] POULIN P., FOURNIER A.: Lights from highlights and shadows. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1992), I3D '92, ACM, pp. 31–38.

[PJ95] PARKER S. G., JOHNSON C. R.: Scirun: A scientific programming environment for computational steering. In *Supercomputing '95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing* (Dec 1995), pp. 52–52.

[PKC*16] PUGMIRE D., KRESS J., CHOI J., KLASKY S., KURC T., CHURCHILL R. M., WOLF M., EISENHOWER G., CHILDS H., WU K., SIM A., GU J., LOW J.: Visualization and analysis for near-real-time decision making in distributed workflows. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (May 2016), pp. 1007–1013.

[PKS*03] PAGE D., KOSCHAN A., SUKUMAR S. R., ABIDI B., ABIDI M.: Shape analysis algorithm based on information theory. vol. 1, pp. 229–232.

[Ple91] PLEMENOS D.: *Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes : le projet multiformes*. PhD thesis, 1991. 1991NANT2041.

[PLF*03] PASCUCCI V., LANEY D. E., FRANK R. J., SCORZELLI G., LINSEN L., HAMANN B., GYGI F.: Real-time monitoring of large scientific simulations. In *Proceedings of the 2003 ACM Symposium on Applied Computing* (New York, NY, USA, 2003), SAC '03, ACM, pp. 194–198.

[Pli95] PLIMPTON S.: Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics 117*, 1 (1995), 1 – 19.

[PPB*05] POLONSKY O., PATANÉ G., BIASOTTI S., GOTSMAN C., SPAGNUOLO M.: What's in an image? *The Visual Computer 21*, 8 (Sep 2005), 840–847.

[PRC81] PALMER S. E., ROSCH E., CHASE P.: Canonical perspective and the perception of objects.

[PRJ97] POULIN P., RATIB K., JACQUES M.: Sketching shadows and highlights to position lights. In *Proceedings of the 1997 Conference on Computer Graphics International* (Washington, DC, USA, 1997), CGI '97, IEEE Computer Society, pp. 56–.

[R C14] R CORE TEAM: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[RBB*11] RUIZ M., BARDERA A., BOADA I., VIOLA I., FEIXAS M., SBERT M.: Automatic transfer functions based on informational divergence. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 1932–1941.

[SCW*13] SLAWINSKA M., CLARK M., WOLF M., BODE T., ZOU H., LAGUNA P., LOGAN J., KINSEY M., KLASKY S.: A maya use case: Adaptable scientific workflows with adios for general relativistic astrophysics. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (New York, NY, USA, 2013), XSEDE '13, ACM, pp. 54:1–54:8.

[SFR*02] SBERT M., FEIXAS M., RIGAU J., VIOLA I., CHOVER M.: Applications of information theory to computer graphics. In *Eurographics* (2002).

[SK32] SANDER F., KRUEGER F.: *Gestaltpsychologie und Kunsttheorie: ein Beitrag zur Psychologie architektonischer Gestalten*. Beck, 1932.

[SLF*11] SECORD A., LU J., FINKELSTEIN A., SINGH M., NEALEN A.: Perceptual models of viewpoint preference. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 109:1–109:12.

[SML06] SCHROEDER W., MARTIN K. M., LORENSEN W. E.: *The Visualization Toolkit (4th Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[SP05] SOKOLOV D., PLEMENOS D.: Viewpoint quality and scene understanding. In *Proceedings of the 6th International Conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage* (Aire-la-Ville, Switzerland, Switzerland, 2005), VAST'05, Eurographics Association, pp. 67–73.

[SPFG05] SBERT M., PLEMENOS D., FEIXAS M., GONZÁLEZ F.: Viewpoint quality: Measures and applications. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2005), Computational Aesthetics'05, Eurographics Association, pp. 185–192.

[SS02] STOEV S. L., STRASSER W.: A case study on automatic camera placement and motion for visualizing historical data. In *IEEE Visualization, 2002. VIS 2002.* (Oct 2002), pp. 545–548.

[SSB13] Serin E., Sumengen S., Balcisoy S.: Representational image generation for 3d objects. *Vis. Comput. 29*, 6-8 (June 2013), 675–684.

[Tau95] Taubin G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision* (June 1995), pp. 902–907.

[TFT05] Takahashi S., Fujishiro I., Takeshima Y.: Interval volume decomposer: a topological approach to volume traversal. In *Visualization and Data Analysis 2005* (2005), Erbacher R. F., Roberts J. C., Grohn M. T., Borner K., (Eds.), vol. 5669, International Society for Optics and Photonics, SPIE, pp. 103 – 114.

[TFTN05] Takahashi S., Fujishiro I., Takeshima Y., Nishita T.: A feature-driven approach to locating optimal viewpoints for volume visualization. In *VIS 05. IEEE Visualization, 2005.* (Oct 2005), pp. 495–502.

[TK01] Tarr M. J., Kriegman D. J.: What defines a view? *Vision Research 41*, 15 (2001), 1981 – 2004.

[TTF04] Takahashi S., Takeshima Y., Fujishiro I.: Topological volume skeletonization and its application to transfer function design. *Graph. Models 66*, 1 (Jan. 2004), 24–49.

[VBP*09] Vieira T., Bordignon A. L., Peixoto A., Tavares G., Lopes H., Velho L., Lewiner T.: Learning good views through intelligent galleries. *Comput. Graph. Forum 28* (2009), 717–726.

[VFSG06] Viola I., Feixas M., Sbert M., Groller M. E.: Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sep. 2006), 933–940.

[VFSH01] Vázquez P.-P., Feixas M., Sbert M., Heidrich W.: Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), VMV ’01, Aka GmbH, pp. 273–280.

[VFSH03] Vzquez P.-P., Feixas M., Sbert M., Heidrich W.: Automatic view selection using viewpoint entropy and its application to image-based modelling. *Computer Graphics Forum 22*, 4 (2003), 689–700.

[VHP*11] Vishwanath V., Hereld M., Papka M. E., Hudson R., Jordan G. C., Daley C. J. A.: In situ data analysis and i / o acceleration of flash astrophysics simulation on leadership-class system using glean.

[VKG05] Viola I., Kanitsar A., Groller M. E.: Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (July 2005), 408–418.

[WDC*07] Weber G. H., Dillard S. E., Carr H., Pascucci V., Hamann B.: Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (March 2007), 330–341.

[WFM11] Whitlock B., Favre J. M., Meredith J. S.: Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization* (Aire-la-Ville, Switzerland, Switzerland, 2011), EGPGV ’11, Eurographics Association, pp. 101–109.

[WS04] Weber G. H., Scheuermann G.: Automating transfer function design based on topology analysis. In *Geometric Modeling for Scientific Visualization* (Berlin, Heidelberg, 2004), Brunnett G., Hamann B., Müller H., Linsen L., (Eds.), Springer Berlin Heidelberg, pp. 293–305.

[WS11] Wang C., Shen H.-W.: Information theory in scientific visualization. *Entropy 13*, 1 (2011), 254–273.

[YRFB14] Yi H., Rasquin M., Fang J., Bolotnov I. A.: In-situ visualization and computational steering for large-scale simulation of turbulent flows in complex geometries. In *2014 IEEE International Conference on Big Data (Big Data)* (Oct 2014), pp. 567–572.

[YWG*10] Yu H., Wang C., Grout R. W., Chen J. H., Ma K.: In situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications 30*, 3 (May 2010), 45–57.