AUTOMATING CAMERA PLACEMENT FOR IN SITU VISUALIZATION

by

NICOLE MARSAGLIA

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Division of Graduate Studies of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 2022

DISSERTATION APPROVAL PAGE

Student: Nicole Marsaglia

Title: Automating Camera Placement for In Situ Visualization

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Hank Childs	Chair
Brittany Erickson	Core Member
Michal Young	Core Member
Ellen Eischen	Institutional Representative

and

Krista Chronsiter	Vice Provost for Graduate Studies

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded March 2022

DISSERTATION ABSTRACT

Nicole Marsaglia

Doctor of Philosophy

Department of Computer and Information Science

March 2022

Title: Automating Camera Placement for In Situ Visualization

Trends in high-performance computing increasingly require visualization to be carried out using in situ processing. This processing most often occurs without a human in the loop, meaning that the in situ software must be able to carry out its tasks without human guidance. This dissertation explores this topic, focusing on automating camera placement for in situ visualization when there is no a priori knowledge of where to place the camera. We introduce a new approach for this automation process, which depends on Viewpoint Quality (VQ) metrics that quantify how much insight a camera position provides. This research involves three major sub-projects: (1) performing a user survey to determine the viewpoint preferences of scientific users as well as developing new VQ metrics that can predict preference 68% of the time; (2) parallelizing VQ metrics and designing search algorithms so they can be executed efficiently in situ; and (3) evaluating the behavior of camera placement of time-varying data to determine how often a new camera placement should be considered. In all, this dissertation shows automating in situ camera placement for scientific simulations is possible on exascale computers and provides insight on best practices.

CURRICULUM VITAE

NAME OF AUTHOR: Nicole Marsaglia

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2022, University of Oregon
Bachelor of Science, Mathematics, *Summa Cum Laude*, 2013, University of Oregon

AREAS OF SPECIAL INTEREST:

Automatic Visualization
Scientific Visualization
High Performance Computing

PROFESSIONAL EXPERIENCE:

Graduate Research Fellow, 2014, 2016 - Present
Graduate Teaching Fellow, 2015, Spring 2020
Graduate Intern, Lawrence Berkeley National Lab, Summer 2019
Graduate Intern, Lawrence Berkeley National Lab, Summer 2018
Graduate Intern, Oak Ridge National Lab, Summer 2017

GRANTS, AWARDS AND HONORS:

Best Paper Award: "A Flexible System for In Situ Triggers." At the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV) 2018
Curtis Scholarship, University of Oregon, 2012

PUBLICATIONS:

Nicole Marsaglia, Meghanto Majumder, & Hank Childs (Jan. 2022). Optimal Viewpoint Placement Over Time (OVPOT) for Scientific Simulations. *(In Progress)*

Nicole Marsaglia, Manish Mathai, Stefan Fields, & Hank Childs (Dec. 2021). Automatic In Situ Camera Placement for Large-Scale Scientific Simulations. *(In Submission)*

Nicole Marsaglia, Yuya Kawakami, Samuel D. Schwartz, Stefan Fields, & Hank Childs (Oct. 2021). An Entropy-Based Approach for Identifying User-Preferred Camera Positions. *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*

Yuya Kawakami, Nicole Marsaglia, Matthew Larsen, & Hank Childs (Nov. 2020). Benchmarking In Situ Triggers Via Reconstruction Error. *Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*

Dave Pugmire, James Kress, Jieyang Chen, Hank Childs, Jong Choi, Dmitry Ganyushin, Berk Geveci, Mark Kim, Scott Klasky, Xin Liang, Jeremy Logan, Nicole Marsaglia, Kshitij Mehta, Norbert Podhorszki, Caitlin Ross, Eric Suchyta, Nick Thompson, Steven Walton, Lipeng Wan, & Matthew Wolf (Aug. 2020). Visualization as a Service for Scientific Data. *Smoky Mountains Computational Sciences and Engineering Conference*

Nicole Marsaglia, Samuel Li, Kristi Belcher, Matthew Larsen, & Hank Childs (June 2019). Dynamic I/O Budget Reallocation For In Situ Wavelet Compression. *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*

Matthew Larsen, Amy Woods, Nicole Marsaglia, Ayan Biswas, Soumya Dutta, Cyrus Harrison, & Hank Childs (Nov. 2018). A Flexible System for In Situ Triggers. *Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*

Samuel Li, Nicole Marsaglia, Christoph Garth, Jonathan Woodring, John Clyne, & Hank Childs (Sept. 2018). Data Reduction Techniques for Simulation, Visualization and Data Analysis. *Computer Graphics Forum (CGF)*

Nicole Marsaglia, Samuel Li, & Hank Childs (Sept. 2018). Enabling Explorative Visualization with Full Temporal Resolution Via In Situ Calculation of Temporal Intervals. *Lecture Notes in Computer Science(Springer)*

Samuel Li, Nicole Marsaglia, Vincent Chen, Christopher Sewell, John Clyne, & Hank Childs (June 2017). Achieving Portable Performance for Wavelet Compression Using Data Parallel Primitives. *EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV)*

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor, Hank Childs, for his unwavering belief in me and immense support over the years. I want to thank my therapist, Zanne Miller, for helping me rebuild my confidence and internalize my successes. And, lastly, I want to thank my family and partner, none of this would have been possible without your love.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

To Odette and Tormund, my favorite people.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

I was the primary author of this manuscript, and Hank Childs provided editorial suggestions.

This dissertation considers automating in situ camera placement on supercomputers. The goal of this research is to enable domain scientists to efficiently and confidently produce quality images visualizing exascale simulations. This research topic requires study in multiple areas, in particular how to predict user preferences and how to run efficiently on supercomputers. This chapter lays out the motivation for this dissertation as well as the research problems it aims to answer. The remainder of this chapter is organized as follows: Section 1.1 covers the basics and current state of high-performance computing (HPC); Section 1.2 details a challenge in HPC and how it affects scientific visualization; Section 1.3 details the importance and use of in situ processing; Section 1.4 enumerates the challenges of automated in situ camera placement; and Section 1.5 details the projects of this dissertation.

1.1 Supercomputing Trends

Scientific breakthroughs in a variety of fields — astrophysics, biophysics, chemistry, medicine, nuclear physics, and more — are coming from computational simulations. These simulations often require significant compute power to obtain accurate results. In many cases, this compute power can only be obtained by using supercomputers, i.e., connecting many regular CPUs and GPUs together via a network and operating the collective computing power in a coordinated manner. Supercomputers are typically measured in FLoating point OPerations per Second, abbreviated "FLOPS." Organizations in the US, China, Japan, and

Europe are currently endeavoring to make the world's first "exascale" machine, meaning a supercomputer that can achieve 10^{18} FLOPS. As regular computers are approximately 10^9 FLOPS, this means these machines will be on the order of one billion times more powerful than a typical desktop computer.

1.2 Supercomputing Challenge: the I/O Gap

The computational power of supercomputers has been steadily trending upwards, reaching petascale capabilities in 2008 with the expectation of reaching exascale in Fall of 2022. This increase in computational power allows researchers to run larger and larger scientific simulations. Because of the increase in compute-per-node, modern supercomputers can generate massive amounts of data in a short amount of time. Unfortunately, the speed in which supercomputers can generate data severely outpaces its ability to save data, leading to a phenomenon referred as the *I/O gap*. And as the gap between I/O and compute capabilities continues to grow on state-of-the-art supercomputers, scientists will have to adapt their workflows for analyzing and visualizing their generated simulation data.

In the traditional paradigm, referred to as post-hoc processing, simulations save out entire time slices of data to permanent storage, and visualization occurs afterwards by reading these time slices from disk. But with the rise of the I/O gap, researchers operating within the traditional paradigm had limited options: either save less often and potentially miss important phenomena, or deal with significant time penalties for storing data — neither of which are great solutions. The shortcomings of the post-hoc paradigm has led to increased interest in *in situ processing*.

1.3 In Situ Processing

In situ processing, i.e., processing data as it is generated, has become an important technique for visualizing and analyzing data from computational simulations on modern supercomputers. Its benefits include reducing I/O costs and access to increased temporal resolution. That said, in situ processing also can generate new challenges that were not present with the traditional model of post hoc processing. In particular, while it is possible to perform in situ processing with a human-in-the-loop (HITL), the large majority of in situ processing occurs with no human-in-the-loop. This change is important: where the HITL model enabled domain scientists to bring their expertise to direct visualizations and analyses, their absence requires new approaches to perform this direction.

There are multiple strategies for directing visualizations and analyses with no human-in-the-loop. One strategy is to determine how visualization and analysis should be carried out beforehand and encoding these directions into the in situ workflow. In particular, sometimes predecessor calculations inform good settings, and those settings can be reused. Another strategy is to defer, i.e., reduce the data set to a small enough form that it can be saved to disk, and then perform the visualizations and analyses afterwards in a post hoc and HITL fashion. That said, this strategy can create a tension between the amount of data reduction that can occur and the loss of data integrity. A third strategy is to automate the settings for visualizations and analyses. In this case, in situ infrastructures must be augmented with new algorithms whose sole purpose is to calculate these settings. For example, an in situ infrastructure would need to contain not only an isosurface algorithm, but also an algorithm for picking the isolevels. We feel all three strategies are useful, and require attention from the in situ community. With this dissertation, we

focus on the third strategy (automation), and more specifically, automating camera placement.

1.4 Challenges of Automatic Camera Placement

Camera placement is a critical task for scientific visualization. In a post hoc setting, the camera placement process typically involves starting with a default camera position (e.g., zoomed out with the camera translated down the Z-axis and pointed at the center of the data set) and a domain scientist using a mouse to modify the camera location to a position or positions that increase their insight. In the context of the automation strategy, in situ systems would then need algorithms for identifying the camera position(s) that are as useful as the ones the domain scientist would produce in a HITL setting. This is a somewhat daunting task, as it requires evaluation of what makes one camera position more useful than another. Fortunately, the scientific visualization community has already done significant research on this topic [104, 27, 48, 118], having developed a number of Viewpoint Quality (VQ) metrics that can evaluate a given camera placement for a data set. That said, these research works have not considered in situ use cases nor scientific data. Instead, this previous research was performed with goals of saving time for domain scientists (since they would not have to go through the process of finding good camera positions themselves) or of improving key camera position (i.e., using automation to become better at finding camera positions than domain scientists, and then helping them find camera positions they would not have discovered on their own).

One important concern regarding existing camera placement algorithms is their application to a parallel setting. All published work to date focuses on serial implementations, and parallelizing serial algorithms can be difficult. In particular, a

number of the algorithms require a large amount of coordination when implemented in a distributed fashion.

A second important concern regarding existing camera placement algorithms are their execution times. For many algorithms, the work to evaluate a camera position is comparable to actually performing a render. That level of slowness suggests different approaches altogether — instead of evaluating N camera positions, why not just render images from those N positions and allow a domain scientist to explore them all? This idea is not just viable, but is further a good idea. It has been pursued by the Cinema project, among others. That said, we feel that automated camera placement still has important roles to play. One potential role is to ensure that there are quality camera positions, i.e., if the N camera positions are all bad, then camera placement algorithms can help inform this and cue in situ infrastructures to try more camera positions. But we believe the most important role is for presentation graphics, i.e., the images that domain scientists use to advertise their own work (journal covers, conference presentations, etc.). In this case, we envision that camera placement algorithms will find the best camera position, and then very high quality renderings will be performed at this camera position. Since these very high quality renderings are expensive to produce (i.e., ray tracing with many secondary rays), the cost of the camera placement will be small in comparison.

1.5 Dissertation Question

This dissertation aims to answer the following question: **is automated in situ camera placement viable for large-scale simulations and what are the best practices?** To do this, we have broken the larger dissertation question into four research subquestions:

- **RQ1:** What viewpoints of scientific data sets do visualization experts and domain scientists want to see and can VQ metrics be used to user predict preference?
- **RQ2:** Can VQ metrics be performed efficiently in situ at scale?
- **RQ3:** Can a quality viewpoint be found quickly at scale?
- **RQ4:** How often should a simulation automate camera placement?

In all, we plan to answer this dissertation's research question by creating camera placement metrics that can be performed efficiently with a low memory footprint at large-scale as well as evaluating methods for finding the best viewpoint.

The remainder of this chapter is organized as follows: Section 1.5.1 describes the research thrust to answer RQ1; Section 1.5.2 describes the research thrust to answer RQ2; Section 1.5.3 describes the research thrust to answer RQ3; and Section 1.5.4 describes the research thrust to answer RQ4.

1.5.1 RQ1: Expert Survey and Oracles Based on VQ Metrics.

This research thrust studies the correlation between user preference and VQ metric score. To do this we will survey a group of visualization researchers and domain scientists who will rank viewpoints from a number of different scientific simulations. These rankings will then be compared to the VQ metric scores to determine if there is any correlation. The results of this survey can provide vital insights into what VQ metrics produce the best results and which VQ metrics may be used in conjunction to yield the best image viewpoints.

The results of this research thrust are detailed in Chapter III.

1.5.2 RQ2: Distributed Implementation and Performance Study.

For this research thrust, we will implement the selected VQ metrics

in situ and conduct a performance study. Specifically, we want to make sure that the VQ metric calculations do not overburden the simulation and that the metrics can be run efficiently on HPC architectures. To guarantee our approach can be computed quickly and achieve portable performance, we plan to integrate the VQ metrics into Ascent [71], a lightweight in situ analysis and visualization framework that is built using the VTK-m library. By implementing our approach into software infrastructures that utilize VTK-m, we are guaranteeing that our approach can achieve high performance on current and emerging architectures. Once implemented within Ascent, we plan to study the execution of the metrics in situ with varying number of camera viewpoint samples and parallelism of varying large-scale scientific simulations on the Cori supercomputer at Lawrence Berkeley's NERSC supercomputing facility.

Prior to beginning this thesis, we already determined which camera placement metrics are fit for in situ. Out of the 26 VQ metrics surveyed for my Area Exam, seven were identified as viable for in situ implementation. These VQ metrics were chosen because of their low communication costs, only needing one or two global coordinations in order to compute a metric score. Additionally, they utilize data that is freely available in the rendering process, meaning they all execute with a small memory footprint. And lastly, these metrics are simple and, to varying degrees, can be calculated at a low computational cost.

The results of this research thrust are detailed in Chapter IV.

1.5.3 RQ3: Search Routine. The next research thrust involves using VQ metrics to identify the best viewpoint over all possible viewpoints. In order to find the best viewpoint, we plan on developing search algorithms to pinpoint quality camera placements without having to consider 100s or 1,000s of viewpoints.

This work explores the benefit or cost of searching for the "best" viewpoint and will study the tradeoffs between the highest obtained VQ metric score and the incurred execution cost for a number of different in situ search budgets (i.e., how many camera placements are considered).

The results of this research thrust are detailed in Chapter IV.

1.5.4 RQ4: Temporal Coherence. The last research thrust will focus on the temporal behavior of the VQ metrics as a scientific simulation evolves. For the most part, scientific data sets are temporally coherent, hence, a simulation may have the same "best" viewpoint for some VQ metric over a period of time. Thus, as the simulation progresses, instead of searching for the best viewpoint again, maybe it is enough to check if the previous best viewpoint is still good.This will also amortize the cost of the initial search, as well as subsequent searches.

And in the case where the scores are sporadic, automated in situ camera placement can be useful for directing researchers to what is important, or to the fact that something has changed. The behavior of these metrics over time will be studied on a number of different scientific simulations and will help influence the usage of VQ metrics, such as frequency of execution.

The results of this research thrust are detailed in Chapter V.

CHAPTER II

BACKGROUND

I was the primary author of this manuscript, and Hank Childs provided editorial suggestions.

2.1 Introduction

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 nor requiring a large 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 [79, 90] and automatic transfer functions for volume rendering [141, 111, 136, 137], to name a few. As already stated in the introduction, the primary focus of this dissertation is automatic camera placement while running in situ.

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 [27]. The majority of the techniques perform some calculation based on the geometry of the data, with few techniques for scientific field data.

This chapter 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.

2.1.1 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 [58, 7, 114].

In the 1930s a mathematician named George D. Birkhoff attempted to quantitatively measure beauty [22, 23]. 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 [130] 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 [25, 97]. Kamada et al. [68] 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. [25], 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.

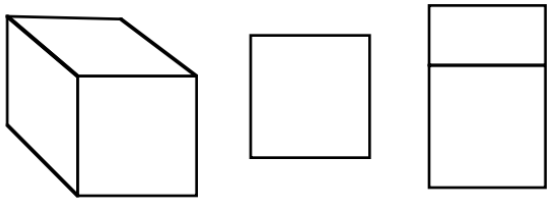


Figure 1. Taken from Barral et al. [13], this image depicts one non-degenerate view of a cube (left) and two degenerate views of a cube (center and right) as defined by Kamada et al. [68]. Notice that the non-degenerate cube is shown at a three-quarter, or canonical view.

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. [29] 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 [67, 105, 106]. Current works are based on inverse lighting techniques, where the optimal light source is deduced from an expected result. In general, most current

techniques are not fully automatic and require user interaction [91, 61]. Gumhold [59] presents an automatic method based on *light entropy*, but results are not consistent.

Bordoloi and Shen [28] 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?"

2.1.2 How to Find the Best Image?. Finding the best image of a data set is difficult problem. For instance, a data set could contain multiple "best images" depending on what the domain scientist wants to convey. Polonsky et al. [104] treats viewpoint selection as a function where the best viewpoint maximizes this function. A function that quantifies a viewpoint is called a *view descriptor*, with later research renaming view descriptor functions as *Viewpoint Quality (VQ) metrics*. Polonsky et al. constructed their VQ metrics 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 domain scientist wants to convey, researchers could utilize one or more of these principles to design a viewpoint quality metric.

2.2 Viewpoint Quality (VQ) 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 2.2.2 surveys the viewpoint quality measures that are based on the geometry of the data. And Section 2.2.3 surveys the viewpoint quality measures that are based on the field data.

2.2.1 Notation. This section will define the notation used for these metrics. Table 1 comes from the in-depth survey from Bonaventura et al. [27] where they compared 22 different viewpoint metrics. The notation developed is based off of an information channel developed by Feixas et al. [54]. 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, t, of the model Z from some viewpoint $v \in V$ is denoted $a_t(v)$.

Feixas et. al [54] 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 $p(z) = \sum_{v \in V} p(v)p(z|v)$.

2.2.2 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.

Work by Bonaventura et al.[27] and Secord et al.[118] 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 chapter, and the remainder of this

dissertation will follow the same naming conventions and categorizations for each measurement.

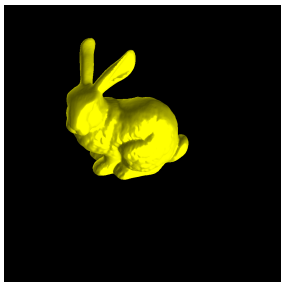
2.2.2.1 Area. This section covers the measurements that involve the projected area of the polygons in relation to a particular viewpoint, including view area, ratio of visible area, and surface area entropy. These metrics are useful on datasets with highly varying polygons and when maximizing visible area is important. They are all relatively quick, unless they involve mutual information, and can be simply implemented with the aid of graphics hardware [13, 14].

Number of Visible Triangles. Several of the first measurements on viewpoint selection came from Plemenos [101] and were then expanded upon by Plemenos and Benayada [102]. 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:

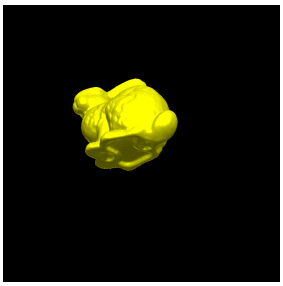
$$VQ_1(v) = \sum_{z \in Z} vis_v(z).$$

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, Number of Visible Triangles.



(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.

Projected Area. Plemenos and Benayada [102] 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.

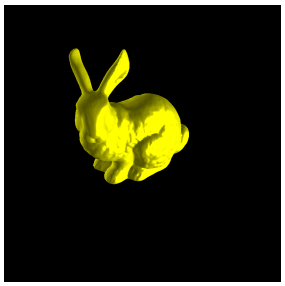
Their second measurement is simply the total visible area of the model from some viewpoint is defined as follows:

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

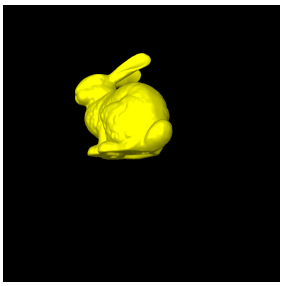
For this measurement, the higher the projected area of the model, the better the viewpoint.

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 VQ_2, Projected Area.

Plemenos and Benayada (PB). The next measurement from Plemenos and Benayada is a combination of $VQ_1(v)$ and $VQ_2(v)$, creating a viewpoint ratio



(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.

based on both the total number of visible triangles and the total projected area. This measurement is expressed as follows:

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

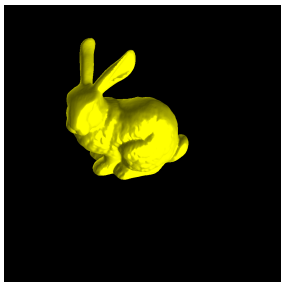
where N is the total number of polygons of the model ($N = |Z|$) and R is the resolution of the image (i.e. total number of pixels).

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.

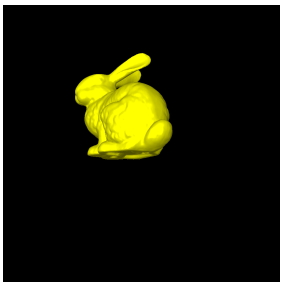
Figure 4 shows the best and worst viewpoints of the Stanford Bunny using VQ_3, PB.

Visibility Ratio. Lastly, Plemenos and Benayada [102] 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 both metrics VQ_3, PB, and VQ_7, VKL.

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

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, Projected Area.

Viewpoint Entropy. To determine the best, most representative viewpoint of an image, a new technique is used involving information theory called Viewpoint Entropy [134, 115].

Viewpoint Entropy is based off of Shannon Entropy [41, 24]. 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. [134] 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. [104] 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 [44] to quantify the information content in the brain in terms of stimuli and response. Bonaventura et al. [26] 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. [116] 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:

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

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.

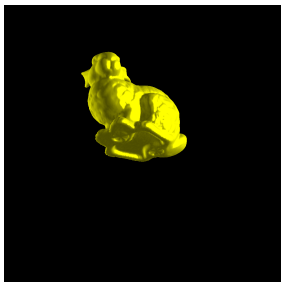
Figure 4 shows the best and worst viewpoints of the Stanford Bunny using this metric.

Viewpoint Mutual Information (I_1). Feixas et al. [54] introduces this viewpoint selection method that quantifies the degree of correlation between the viewpoints and set of polygons. I_1 is expressed as follows:

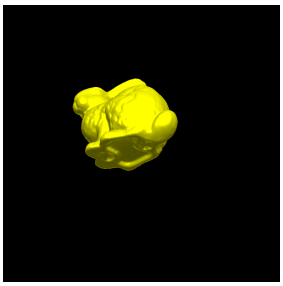
$$VQ_8(v) = I_1(v; Z) = \sum_{z \in Z} p(z|v) \log \frac{p(z|v)}{p(z)}.$$

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 [44].

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.



(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, Viewpoint Mutual Information.

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

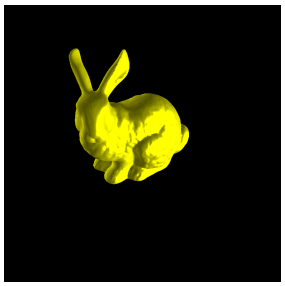
Information I_3. Again from neuroscience, Butts [31] presents a metric to quantify the information associated with a stimulus using mutual information. Whereas Bonaventura et al. [26] proposed this metric as a viewpoint quality measure. I_3 is expressed as follows:

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

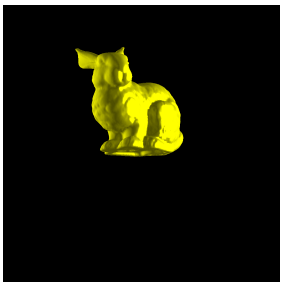
such that

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

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."[27]



(a) The best viewpoint of the Stanford Bunny.



(b) The worst viewpoint of the Stanford Bunny.

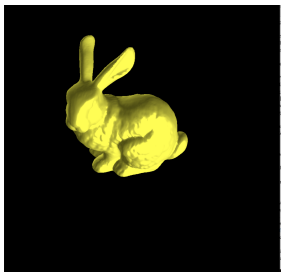
Figure 6. The best and worst viewpoints of the Stanford Bunny determined by metric VQ_9, Information I_3.

Figure 6 shows the best and worst viewpoints of the Stanford Bunny using VQ_9, I_3.

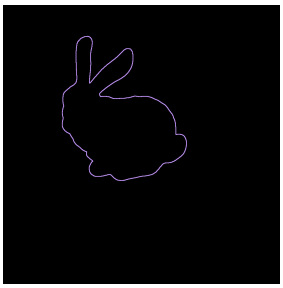
2.2.2.2 Silhouette. This section surveys metrics that utilize a model's silhouette. The silhouette, or *occluding contour*, of an object is a view-dependent metric. Simply, the silhouettes of an object are the edges that are created if the object were to be represented as a single color on a plain background, 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 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. [104]. 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:



(a) The Stanford Bunny.



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

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

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

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. [96] was the first to combine silhouettes and entropy, and it was Polonsky et al. [104] 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. [135] introduced a new metric, based on the work by Felldman et al. [55], 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. [118] 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.

2.2.2.3 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 Stasser[124] 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:

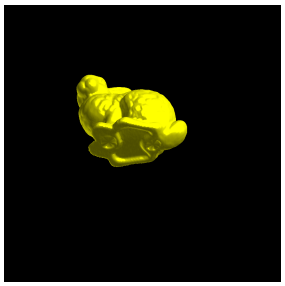
$$VQ_{14}(v) = \alpha p(v) + \beta d(v) + \gamma(1 - |d(v) - p(v)|),$$

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 $\gamma = \frac{1}{2}$.

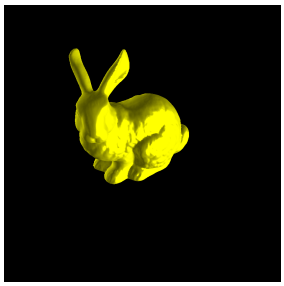
For this metric, the highest value will correlate to the best viewpoint. This metric is best used when projected area as well as depth are key components of the model, such as with terrain data sets. This metric is adaptable and allows for some user input in terms of prioritizing depth, area, or both.

Maximum Depth. Secord et al. [118], inspired by Stoev and Straber, also considered depth when creating a viewpoint metric. Their first metric is simply the maximum depth of the model. Maximum depth is defined as follows:

$$VQ_{.49}(v) = depth(v),$$



(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 VQ_{16}, Depth Distribution.

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. [118] 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.

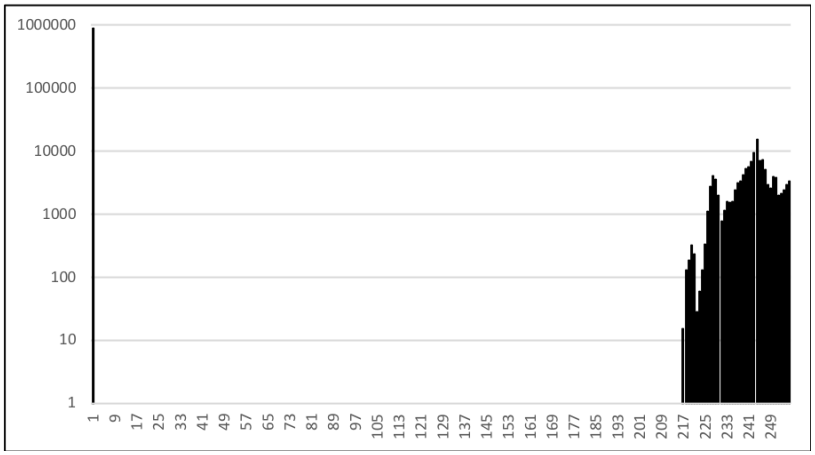


Figure 9. The normalized depth histogram corresponding to Figure 7a. The Depth Distribution metric will favor viewpoints whose histograms are evenly distributed with few peaks or valleys. This analysis used 256 bins for the 1M pixels of the image.

2.2.2.4 ***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. 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 [28] 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 [30]. They considered the Kullback-Leibler difference [24] as well, but two images with differing occlusions cannot be compared with that measure.

Feixas et al. [54] expanded on this research and used instability as a metric for image-based rendering. And Lin [82] 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:

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

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. [134] 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 esthetically important.

2.2.2.5 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. [96], interested in shape analysis, first proposed calculating the entropy of the Gaussian curvature distribution over the entire surface of the object. Polonsky et al. [104] 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. [75] present a metric for computing the mesh saliency of a 3D object. This is based on the center-surround method from Itti et al. [63] which computed the saliency for 2D images.

To calculate visible saliency, the curvature at every vertex is calculated using the strategy presented by Taubin [131]. 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.

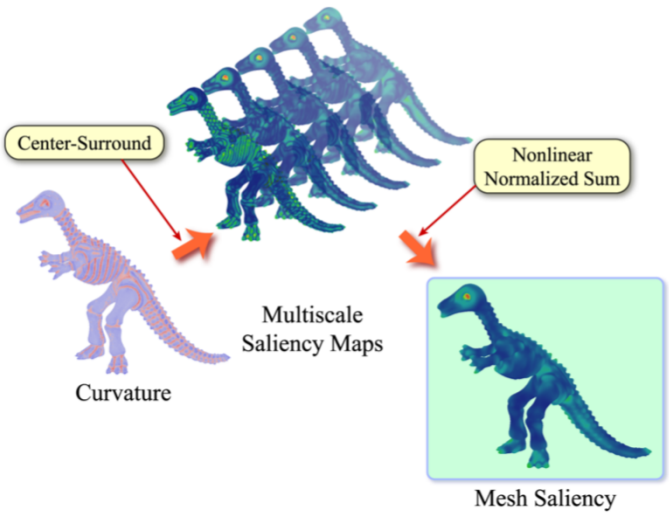
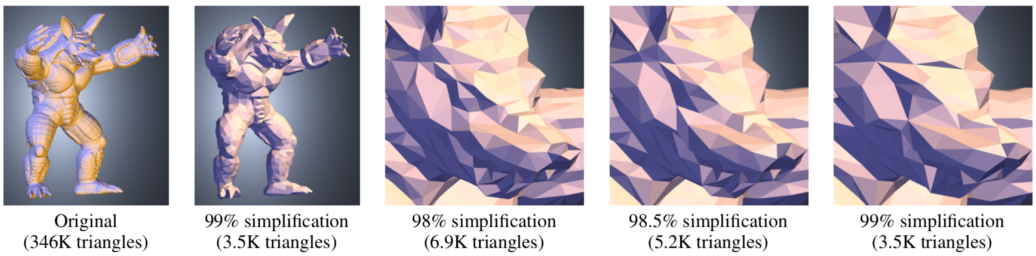


Figure 10. To compute Visible Saliency, the mean curvature is computed for each vertex. The vertex saliency is then computed as the difference between mean curvatures filtered with a narrow and broad Gaussian. Multiple saliency maps are computed by varying σ, the Gaussian standard deviation. Lastly, the saliency of the viewpoint is the aggregate of all the saliency maps using a non-linear normalization. Taken from Lee et al. [75].

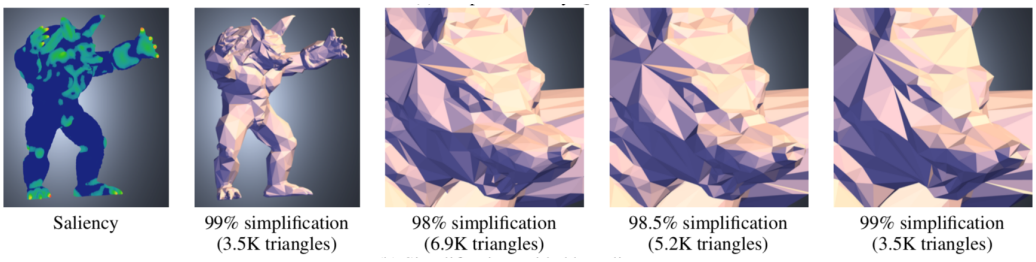
Sokolov and Plemenos [122] also implemented this metric, but for curvature they used the standard angle deficit approximation K_x, as seen in VQ_{19}.

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

Projected Saliency. Feixas et al. [54] apply the idea of mesh saliency to individual polygons. The saliency of a polygon is defined as the average



(a) Mesh simplification using QSlim [56].



(b) Mesh simplification guided by the saliency map.

Figure 11. Taken from [75], 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.

dissimilarity between this polygon and its neighbors. This metric is based on mutual information, similar to metric VMI from Section 2.2.2.1.

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. [54] extends VMI from Section 2.2.2.1 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. [119] 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.

2.2.3 Data Driven Quality Measures.

This section will cover all VQ metrics that are based on field 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 [140] survey the use of information theory in scientific visualization.

2.2.3.1 Entropy.

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

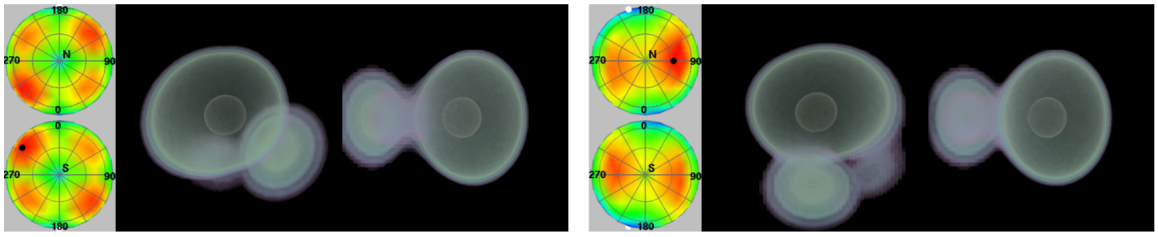
Viewpoint Entropy. The first metric is to utilize viewpoint entropy established by Vazquez et al. [139, 134]. Viewpoint entropy would maximize the visible entropy of each face. Unfortunately, a regular mesh has at most six faces.

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.

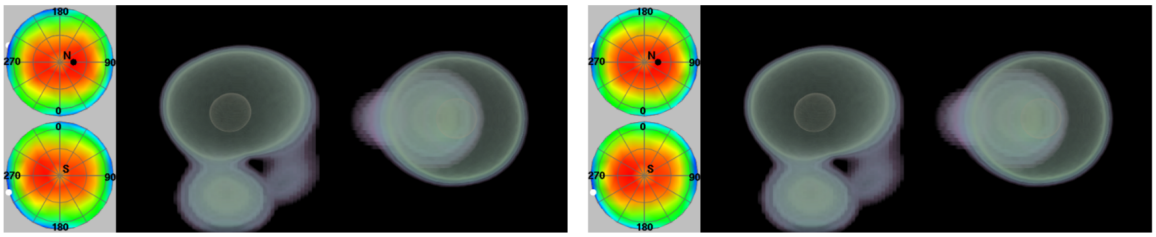
Isosurface Entropy. Takahashi et al. [127] applied viewpoint entropy to volumetric data, utilizing the inherent geometry that is based on the transfer function. Let $p_i(i = 0, \ldots, n-1)$ be the set of scalar values that has been uniformly sampled from the entire data set. This set of scalar values 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:

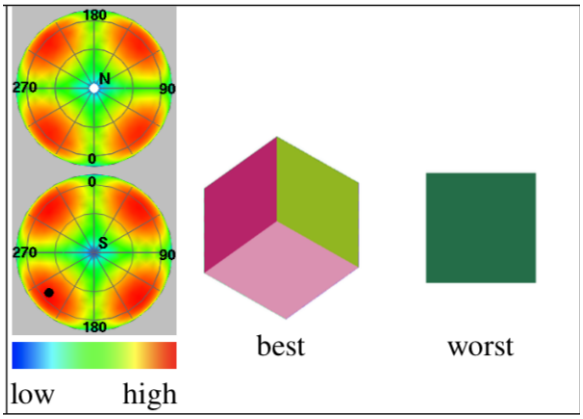
$$E_i(v) = \frac{1}{\log(m_i + 1)} \sum_{j=0}^{m_i} \frac{a_{ij}}{R} \log \frac{a_{ij}}{R},$$



(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 Entropies (12a) and Interval Volume Entropies (12b) metrics, and respective key (12c). Taken from [127].

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 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. [127] 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 [128, 142].

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.

Then the weighted isosurface entropy is defined as follows:

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

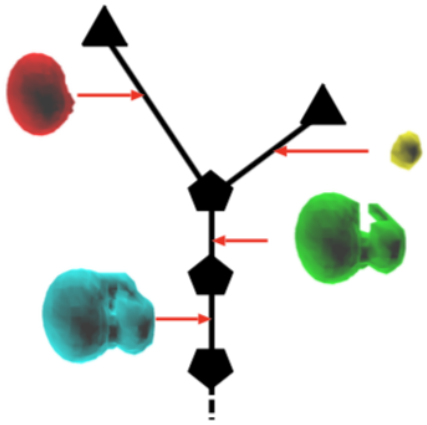


Figure 13. An example of a contour tree and the respective interval volumes associated with sections of the tree. Taken from [127].

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. [127] 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) [126] to represent an interval volume as a contour tree [12], 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. [127] 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 $L = \sum_{i=0}^{n-1} \lambda_i$.

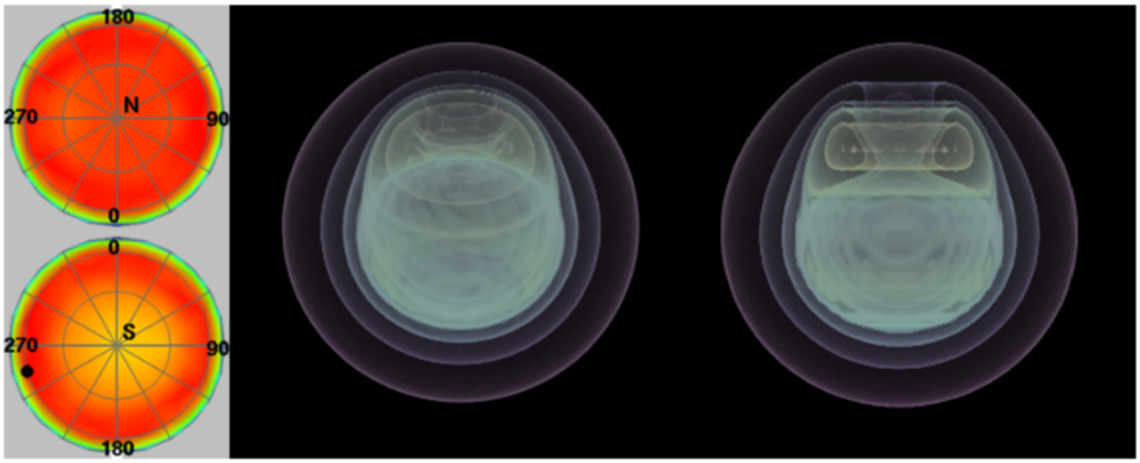
The best viewpoint will have the highest entropy.

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.

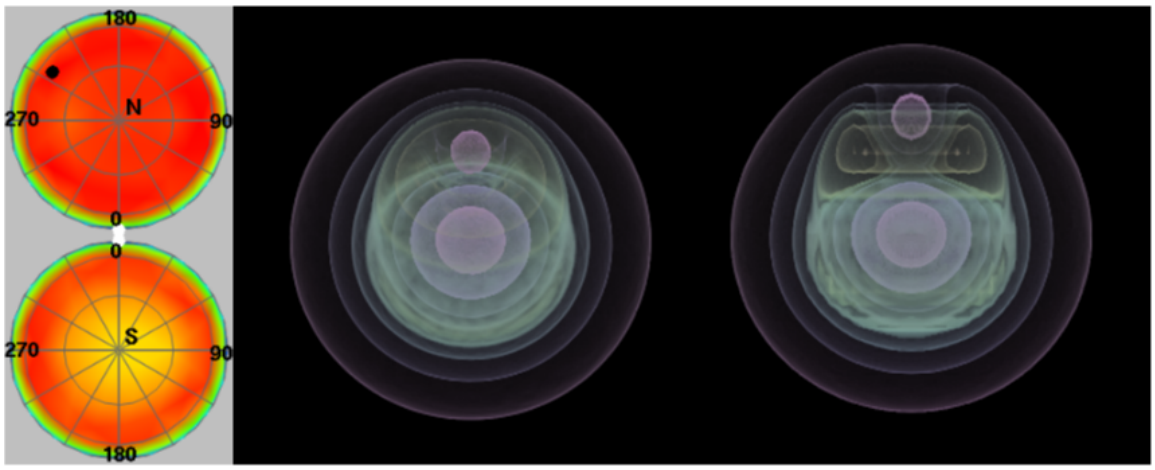
As a final note, for extremely large data sets, these metrics are computationally heavy with potentially high communication costs. That said, utilizing software from Section 2.3 can help mitigate these performance costs.

2.3 In Situ Analysis and Visualization Software

This section surveys notable 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.



(a) Weighted Interval Volume Entropy using a single-dimensional transfer function.



(b) Weighted Interval Volume Entropy 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 Entropy 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 [127].

ADIOS. The Adaptable IO System (ADIOS) [84] 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 [95] is a many-core implementation of the Visualization Tool Kit (VTK) [117]. 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 [73, 81, 77, 69, 80, 94, 147, 108, 78]. The VTK-m library plays a critical role in many of the following software implementations.

Ascent. Developed by Larsen et al. [71], Ascent is a multi-institutional project funded by the Exascale Computing Project (ECP) [93]. A later implementation of Strawman [72, 62], Ascent is a flyweight infrastructure that allows for in situ analysis and visualization of scientific datasets. Ascent utilizes VTK-m [95] 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 [109]). 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 [3] is 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 [35] 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 [143] is a tool in VisIt that facilitates in situ visualization. This allows

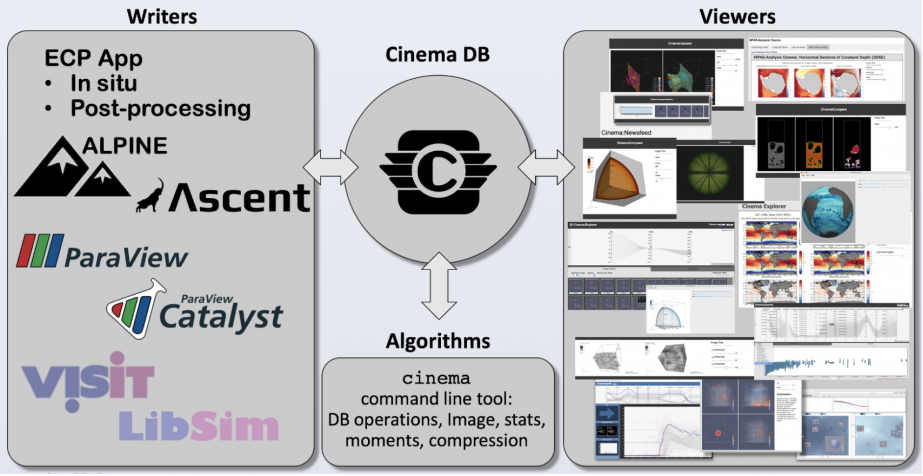


Figure 15. The Cinema ecosystem is composed of databases, algorithms, writers, and viewers. Cinema databases have been integrated as an export option for a number of state-of-the-art in situ analysis and visualization software, or writers. Cinema can employ sophisticated quantitative and qualitative algorithms on the database. The output can then be visualized using the Cinema viewer either post-hoc or in situ. Taken from [47].

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 [2] is an open source, data analysis and visualization application that utilizes VTK [117] 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 [9] is a library within Paraview that allows for in situ visualization and analysis.

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

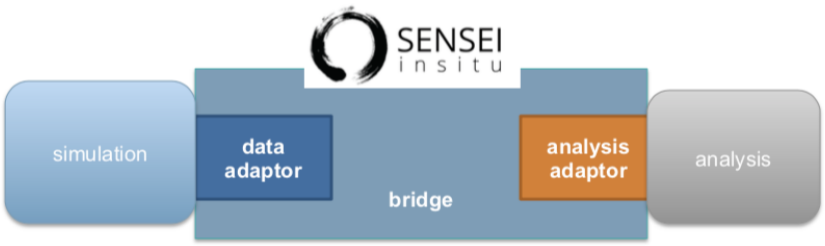


Figure 16. SENSEI is a generic interface that provides a bridge to many technologies. With a simple API, the user can decide which analysis routine to run. In other words, the user can "write once and run every where." Taken from [36].

SENSEI. SENSEI [10] 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 is are the main 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

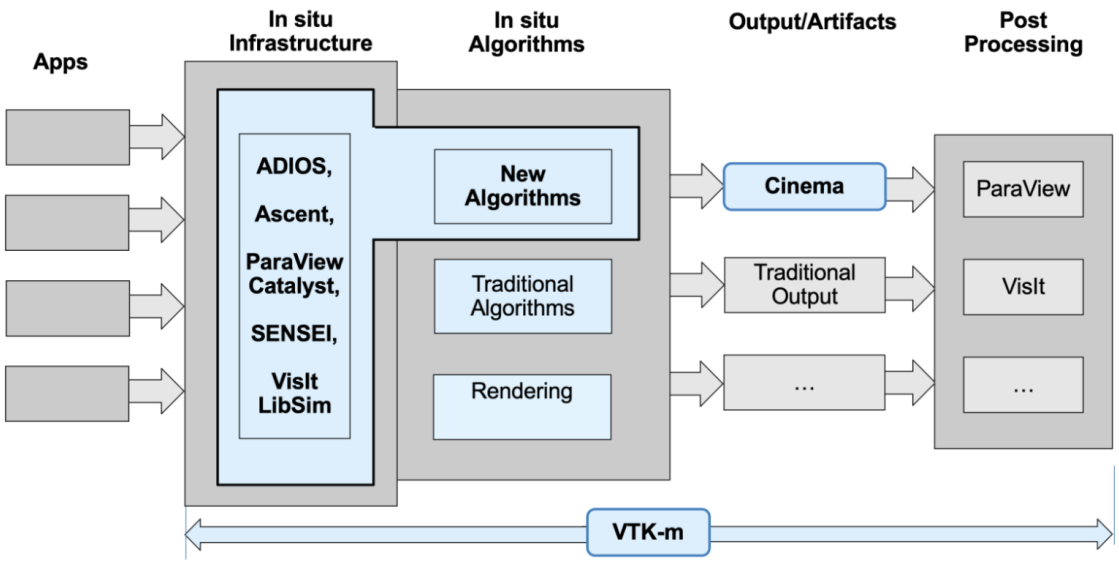


Figure 17. The current state-of-the-art for in situ analysis and visualization software and libraries. This diagram depicts the workflow from the simulation, to the in situ application, I/O routines, and then post processing. Taken from [47].

applications, proving its usefulness and necessity as supercomputers push for exascale.

2.4 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. These works are far from an exhaustive survey, but we feel they are representative of the diversity of practices currently being used.

Combustion.

Yu et al. [149] 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. [17] 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 4,096 cores S3D simulation of one billion cells, their approach had an average wall clock 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. Much like their previous work [149], their camera placement is based on previous small scale runs.

Tokamak.

Work by Pugmire et al. [107] shows that near-real-time visualization and analysis is possible within a distributed workflow. Their work lies in the cross-hairs of simulation monitoring and steering and of in situ visualization and analysis.

There has been a lot of effort into developing steering and monitoring techniques for visualizing data across a network [20, 19, 21, 99, 98], as well as work for in situ approaches [37]. In particular, this work focuses on in transit methods, where data being produced from the XGC1 [34] simulation running 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 [84], a middleware system that has a variety of different transport methods, including Dataspaces [45], FlexPath [43], and ICEE [38]. 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. [51] 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. [121] 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 is 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. [89] 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 are four-fold [89]:

– Formulation of an 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 [103] and FLASH [138], 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. [86] integrate Catalyst [9] into Code_Saturn [6], 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 Ivanhoe, 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. [148] integrate Catalyst into the simulation Parallel Hierarchic Adaptive Stabilized Transient Analysis (PHASTA) [64], 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.

2.5 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 timesteps from the following ECP datasets:

- **ExaSky-Nyx** a cosmological simulation: three different timesteps with 55,544 (ExaSky #1), 143,059 (ExaSky #2), and 19,280 (Exasky #3) triangles.
- **ExaAM Truchas** a metallurgy casting simulation: three different timesteps with 21,255 (ExaAM #1), 6,474 (ExaAM #2), and 18,473 (ExaAM #3) triangles.

– **ExaConstit** a metallurgy simulation: two different timesteps with 938,862 (ExaConstit #1), and 135,109 (ExaConstit #2) triangles.

Table 2 shows the average execution time for each metric per viewpoint for each timestep. 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 A.43-A.50 show the best and worst images for each metric for the eight timesteps. 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.

Our conclusion from this mini-study is that no one metric will find the best image, but maybe a combination of metrics can. We believe an entropy calculation on the field data in conjunction with geometric metrics is a promising direction for generating scientifically important image.

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. Mathematical notation for the defined VQ metrics. This notation was developed in the survey by Bonaventura et al. [27].

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.

CHAPTER III

AN ENTROPY-BASED APPROACH FOR IDENTIFYING USER-PREFERRED CAMERA POSITIONS

This chapter is published co-authored work [92]. Contributions to this work are the following: I was the first author on this work. I wrote the majority of code for this project, including implementing and evaluating the metrics, and developing the website. I attained IRB approval for the user study and conducted the user survey with a presentation at the DOE's 2021 Computer Graphics Forum. I was also the primary author of this work. Yuya Kawakami aided in developing and testing the use survey as well as providing statistical expertise for analyzing the survey results. Yuya also helped in editing the manuscript. Samuel D. Schwartz aided in developing and testing of the user survey as well as providing statistical expertise for analyzing the survey results. Samuel implemented the database that recorded and stored the responses to the user survey. Samuel also helped in editing the manuscript. Stefan Fields aided in developing and testing the user survey. Hank Childs aided in developing and testing of the user survey as well as providing statistical expertise for analyzing the survey results. Hank also provided extensive feedback over the course of this work and helped in editing the manuscript.

3.1 Introduction

As discussed in Chapter II, significant number of research investigations have considered using viewpoint quality (VQ) metrics to evaluate the quality of a camera placement. These metrics are designed to produce "better" views as the metric values increase. That said, the evaluation of these metrics has been limited to non-scientific data sets and the metrics have not considered in situ constraints. With this work, we fill this gap by conducting a user study of large

data visualization practitioners using isosurface imagery. Further, while we do not apply our results in situ, we do constrain the metrics considered to those that can be efficiently calculated in such a setting.

As a separate contribution, we introduce new VQ metrics that are based on entropy. We introduce three such metrics, one that measures data entropy, one that measures depth entropy, and one that measures shading entropy. Many previous research efforts have considered entropy for optimizing visualization parameters; the novelty in our work is in the specific form for this specific problem. We also introduce a mechanism for combining metrics. Our findings show that these metrics perform better than comparator VQ metrics.

Summarizing, this work has two significant contributions:

- We introduce new VQ metrics that are appropriate for an in situ setting and demonstrate that these metrics perform better than existing metrics.
- We conduct the first ever user study devoted to large, scientific visualizations, and use the results to evaluate the efficacy of ten VQ metrics.

3.2 Our Method

This section details our method for constructing an oracle that can use VQ metrics to predict user preference. There are two main concepts in this section: (1) how to construct an oracle from VQ metrics? and (2) which VQ metrics do we incorporate into oracles? The first concept is discussed in Section 3.2.1. The second concept is discussed in two parts: Section 3.2.2 discusses new entropy-based VQ metrics that we introduce in this work and Section 3.2.3 discusses existing VQ metrics that we use as comparators.

3.2.1 Constructing Oracles from VQ Metrics. A VQ-based oracle uses VQ metric values to predict which camera a user would prefer. In an in situ

setting, they could be used to automate camera placement: evaluating camera positions and ultimately selecting the one thought to best match user preference. However, in this study, oracles are used in a more limited way: to evaluate which VQ metric best matches user preference. Further, in this study, an oracle considers two camera positions and attempts to predict the user's preference between the two. That said, oracles could trivially be expanded to deal with more than two camera positions.

For the version of oracle we consider, the only inputs are VQ metrics. They do not have access to image data, information about the camera position, or the geometry being rendered. For example, one of the VQ metrics we consider is "Visibility Ratio" (described in Section 3.2.3.4) and one of the oracles we construct attempts to predict user preference using only Visibility Ratio. In this case, the only input to the oracle would be the values for Visibility Ratio for the two camera positions.

As shown in Figure 18, oracles can operate using only a single VQ metric or multiple VQ metrics. The key distinction between them is that single-metric oracles do not need to combine metrics.

A single-metric oracle is an oracle that uses only one metric to make decisions. We created 10 single-metric oracles, one each for our three new entropy metrics (Section 3.2.2) and one each for the seven existing metrics (Section 3.2.3). If a metric M produces score $M(C_1)$ for camera position C_1 and score $M(C_2)$ for camera position C_2 and if $M(C_1) > M(C_2)$, then the oracle would select C_1. For all of the metrics, "bigger is better," so their corresponding oracles choose the higher values. Further, it was not necessary to decide how to deal with equal metric values, since this did not occur in our experiments.

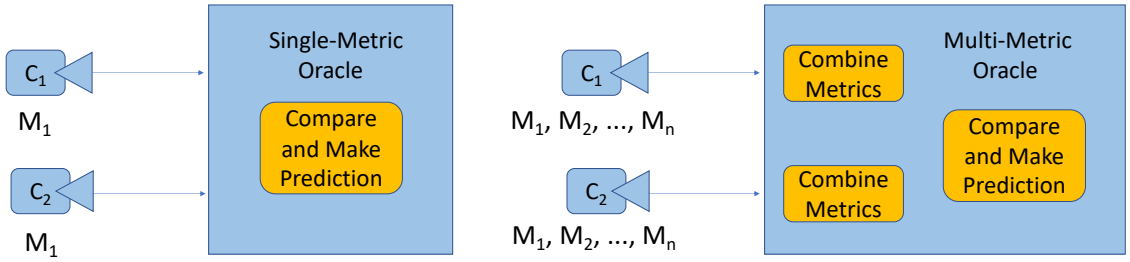


Figure 18. A single-metric oracle (Left) and a multi-metric oracle (Right). Single-metric oracle: For two cameras, C_1 and C_2, and a single metric, M_1, the oracle will compare each camera's respective metric score and determine the best viewpoint. Multi-metric oracle: For two cameras, C_1 and C_2, and n metrics, $M_1, \ldots, M_n$, the oracle will combine each camera's metric scores, compare the combined scores, and determine the best viewpoint.

An oracle that uses multiple metrics has the potential to make better decisions by drawing on different types of information. Given a multi-metric oracle that uses n metrics, $M_1, \ldots, M_n$, these metrics produce tuple $(M_1(C_1) \ldots, M_n(C_1))$ for camera position C_1 and tuple $(M_1(C_2), \ldots, M_n(C_2))$ for camera position C_2. That said, these tuples are not immediately useful, as the metrics produce values with disparate ranges and a variety of units — combining the metrics to make a binary decision is a a fundamental issue with the multi-metric approach. Our solution is to try three different methods for combining metric scores and evaluate all three. That said, looking ahead to results, all three combination methods produced similar findings.

The three methods we consider for combining metrics are:

– **NORM**: Normalizing the scores based on minimum and maximum values and adding the normalized scores together.

– **TIER**: Clustering the scores into tiers using an automated method (Jenks natural breaks optimization [65]) and then adding the tiers together.

– **NONE**: Adding the raw scores together.

The **NONE** approach is not appropriate in almost all cases, but it is appropriate in the case of adding together three entropy scores.

Once the scores were combined, they were compared, and the oracle selected the C_i with the highest value. The **TIER** approach did have ties in some cases, and in these cases a random camera was chosen.

3.2.2 New VQ Metrics. Each of our new metrics utilizes Shannon Entropy [24], which calculates the average level of information. Given a discrete random variable X, with the possible outcomes $x_1, \ldots, x_n$, occurring with the respective probabilities $P(x_1), \ldots, P(x_n)$ the entropy of X is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log(P(x_i))$$

The higher the entropy the more information that is present.

Entropy-based VQ metrics can be constructed by placing fields on images in addition to the typical colors. For example, graphics libraries often produce depth information for the Z-buffer algorithm, and this data augments the image. This depth information can then be used as the discrete random variable for the Shannon entropy calculation, i.e., construct a discrete random variable made up of the depth information for every pixel where data appears and then calculate Shannon entropy on that random variable. Further, maximizing the score of a given entropy-based VQ metric equates to maximizing the information present in the image, at least with respect to its type of data (for example, maximizing entropy in depth information).

We pursued three types of data entropy-based VQ metrics:

– Field Data: the visible data of some user specified field

– Depth Data: the distance from the camera to the visible field data

– Shading Data: the shading coefficients for the visible geometry of the data

These metrics correspond to readily available quantities during the rendering process (field value, depth value, and normal value, which becomes a shading value).

In all cases, we considered the "visible" data (i.e., visible field data, visible depth data, or visible shading data). This means that a scene is rendered, an image is produced, and the data from that image (field, depth, shading) is extracted. If an image has N pixels, and if M pixels have no data occupying that pixel (i.e., background color), then the visible data comes from the $N - M$ pixels that do overlap with the geometry. Considering the example of shading data on an image of 10 pixels ($N == 10$), if 3 pixels have no data ($M == 3$), then the visible shading data would be the set of shading values from the remaining 7 pixels ($N - M$), e.g., $\{0.3, 0.2, 0.6, 1.0, 1.0, 0.8, 0.2\}$.

3.2.2.1 Data Entropy. Data Entropy calculates the entropy of the visible field data from a given viewpoint v. Given field data F, let $F(v)$ be the visible field data for some viewpoint v with elements $f_1, \ldots, f_n$, then Data Entropy is defined as:

$$H(F(v)) = -\sum_{i=1}^{n} P(f_i) \cdot \log(P(f_i))$$

Past research failed to develop this metric because they were primarily focused on developing viewpoint quality metrics for 3D objects that only have geometric data and lack any field data, unlike scientific data. And while Data

Entropy can be applied to any field mesh, it is best if the geometry is unstructured and produces amorphous shapes, as opposed to a three-dimensional rectilinear mesh.

3.2.2.2 Depth Entropy. Depth Entropy calculates the entropy of the distances from the camera to the visible field data from a given viewpoint v. Let $D(v)$ be the set of distances from the camera to the visible field data for some viewpoint v with elements $d_1, \ldots, d_n$, then Depth Entropy is defined as:

$$H(D(v)) = -\sum_{i=1}^{n} P(d_i) \cdot \log(P(d_i))$$

Depth entropy is similar to Secord et al.'s [118] metric, Depth Distribution, which deals with the normalized histogram of depth bins. Finally, this metric is readily applicable to surface data, which fits our isosurface-centric study. Volume rendering would require extending this metric, e.g., adapting for regions of high opacity or when the opacity along a ray hits a threshold.

3.2.2.3 Shading Entropy. Shading Entropy calculates the entropy of the visible shading coefficients. This metric determines the shading coefficient for each visible triangle and then calculates the entropy.

Let $G(v)$ be the set of visible shading coefficients from a viewpoint v, with elements $g_1, \ldots, g_n$, then Shading Entropy is defined as:

$$H(G(v)) = -\sum_{i=1}^{n} P(g_i) \cdot \log(P(g_i))$$

We used flat shading in our calculations, since that was straightforward in our infrastructure, but we note that vertex shading is also possible. Additionally, our infrastructure uses a "miner's light" that is always located above the camera.

And while this work is the first to use shading entropy for viewpoint selection, this metric was first proposed by Gumhold [59] who used shading entropy to determine optimal placement for light sources.

3.2.3 Comparators: Existing VQ Metrics. We consider seven existing VQ metrics as comparators. There are other VQ metrics beyond these seven, but we are only interested in those that can be extended to run both in an in situ setting and in a distributed-memory parallel setting. In all, we only considered a VQ metric if it met three requirements:

- The metric should have a small memory footprint.
- The metric should have a fast execution time.
- The metric should require minimal communication.

The remainder of this section describes the seven VQ metrics, first defining how the metric works and then discussing its merits. These metrics have previously been defined in Chapter II, but are restated for clarity. All descriptions use the notation summarized in Table 1.

3.2.3.1 Number of Visible Triangles. This metric, developed by Plemenos [101] and then expanded upon by Plemonos and Benayada [102], is based on the number of visible triangles from some viewpoint. The best viewpoint will be the one with the highest number of visible triangles. Formally:

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

In the worst case, this metric favors quantity over quality and may choose viewpoints that contain a lot of polygons but little content.

3.2.3.2 Projected Area. This metric, developed by Plemenos and Benayada [102], favors viewpoints that show the most projected area of the data model. This metric simply sums the visible area of the data's geometry. Formally:

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

In the worst case, this metric could select an image that favors one large polygon. Further, maximizing the projected area of the data could also maximize the number of occlusions.

3.2.3.3 Plemenos and Benayada. This metric, from Plemenos and Benayada [102], is a combination of their first two metrics and is defined as follows:

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

Correcting the downside from the first two metrics, Plemenos and Benayada developed a metric that maximizes the number of visible triangles as well as the resolution of the rendered image. While this metric is an improvement, it is susceptible to favoring viewpoints with large occlusions since this metric can be dominated by visible surface area.

3.2.3.4 Visibility Ratio. This final metric from Plemenos and Benayada [102] is the ratio of the real visible surface area over the total real surface area (i.e. the areas in World Space) and is defined as follows:

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

Note that this metric uses the world space geometry rather than device space geometry. This metric has similar advantages and disadvantages as the second metric, VQ_2.

3.2.3.5 *Viewpoint Entropy.*

This metric was first applied to viewpoint selection by Vázquez et al. [134]. Their metric alters Shannon Entropy[41, 24] to take into account the projected area of the scene when centered at a particular viewpoint. Viewpoint Entropy 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. This metric will work best on data sets with varying polygonal size, since larger polygons are penalized in comparison to smaller polygons. An important drawback of this metric is that it will go towards infinity with finer mesh resolutions.

3.2.3.6 *Viewpoint Kullback-Leibler Distance (VKL).*

Developed by Sbert et al. [116], this metric measures the Kullback-Leibler distance between the normalized distribution of the projected areas of polygons from a given viewpoint and the normalized distribution of the real areas of polygons. It is defined as:

$$VQ_6(v) = \sum_{z \in Z} \frac{a_z(v)}{a_t(v)} \log \frac{\frac{a_z(v)}{a_t(v)}}{\frac{A_z}{A_t}}.$$

The best viewpoint, which corresponds to the minimum value, is achieved when the normalized distribution of the projected areas is equal to the normalized

distribution of real areas. In order to make the metrics all follow a "bigger is better" pattern, we multiply this value by -1.

3.2.3.7 Maximum Depth. This metric was defined by Stoev and Strasser [124], but was applied by Secord et al. [118] as a VQ metric. This metric is defined as follows:

$$VQ_7(v) = depth(v),$$

where $depth(v)$ is the maximum depth of the model from some viewpoint v. Depth can be a useful metric for terrain data sets. Terrain data sets are often viewed from above when information is maximized, making the data appear flat, thus it is important to take depth into consideration.

3.3 Corpus for Comparing Viewpoints

This section describes our data corpus for evaluating our method. The corpus is made up of multiple elements. First, the corpus contains images and meta-data about these images. This aspect of the corpus is discussed in Section 3.3.1. Second, the corpus contains results from a user survey on preferred images. This aspect of the corpus is discussed in Section 3.3.2.

3.3.1 Generating a Database of Images. Our corpus considers multiple camera positions for multiple scientific data sets. For each (camera, data set) pair, the corpus contains an image and all VQ metric scores for that image. In all cases, the visualization was of a multi-level isosurface. The remainder of this section describes more detail on the data sets used, the selection of isovalues, and how cameras were placed.

3.3.1.1 Data Sets. A gap in prior research is the lack of application to scientific data sets. To fill this gap, we chose ten large-scale scientific data sets, drawing the IEEE Visualization Conference's Scientific Visualization Contest and

from data sets from the Exascale Computing Project from the United States' Department of Energy. One of our primary goals in selecting these data sets was to consider diverse application domains and diverse imagery, so our results would be (as much as possible) applicable to a larger proportion of scientific data sets.

Four data sets were from the Scientific Visualization Contest:

- Asteroid: A data set of a deep water impact of an asteroid [100].
- Fluid Dynamics: A fluid dynamics data set that models a cylindrical flow of water [70].
- Hurricane: A weather data set of Hurricane Isabel [1].
- Mantle: An earth sciences data set that models the Earth's mantle [120].

Six data sets were from the Exascale Computing Project:

- Constit: A material sciences data set that probes the deformation response of polycrystalline materials [32].
- ExaAm Truchas: A materials science data set that looks at effects within micro-structures of Additive Manufacturing (AM) [16].
- ExaSky Nyx: A cosmological data set that looks at gas dynamics [5].
- Miranda: A hydrodynamics data set of large-scale turbulence [39].
- S3D-N2: A combustion data set of field data N2 [132].
- S3D-UVEL: A combustion data set of field data U Velocity [132].

For each of the chosen data sets we selected a single timeslice we felt was representative of the simulation.

3.3.1.2 Choosing Isovalues.

Each of the data sets are three-dimensional and volumetric, and to each we applied isosurfacing as our visualization operation. Six isovalues were selected, specific to each data set. Our process for choosing isovalues was as follows. Initially, default isovalues were chosen and the data was rendered. If the resulting image is considered "good," we keep those values. Otherwise, we explored the data set to choose isovalues that are "good." "Good" was taken to mean "not bad," as in: the isosurfaces occupied a significant portion of the possible volume and (within reason) the isosurfaces were not bunched together with little separation.

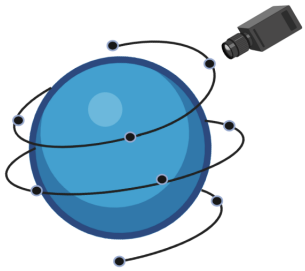


Figure 19. Example of using a Fibonacci Lattice to equally space cameras around a data set. This research used this method determine the ten camera placements for the user survey data sets.

3.3.1.3 Camera Placement.

For each data set, we rendered ten images from ten different viewpoints that can be seen in Figure 20. The camera placements were chosen using Fibonacci's Lattice, a formula that equally spaces points around a sphere, as shown in Figure 19.

We experimented with many camera placement techniques and also with the number of camera positions to include in the survey. We felt the Lattice approach and this number of views provided a nice compromise between two factors. First, we felt the camera positions covered the space of all possible camera positions well — every feature was covered by at least one image. Second, we felt that a

small number of camera positions was beneficial, so we could investigate issues like participant disagreement; if the number of views is very large, then it becomes less likely to get two different participants considering the same pair.

3.3.2 User Study. Our user study collected participant preferences on camera position. The participants in the survey were attendees of the 2021 United States' Department of Energy's Computer Graphics Forum, which is made up large data visualization practitioners across a wide variety of simulation domains. Participants were instructed to make their decision around one central question: "if you had to pick only of these images to represent this simulation, which would it be?"

To take the survey, participants accessed a website where they were presented with a sequence of questions. Each question was composed of two images from the same data set, as shown in Figure 21. Participants were asked to spend 10 minutes answering questions, though they could stop whenever they wanted. Participants answered the question by selecting the image they felt was most representative, or neutral. Having answered, a new pair of images is generated for that user to compare. The questions are randomly generated on demand for each user. To generate a question, first the data set was randomly chosen, then the two images to be compared ware randomly chosen, making sure to not to select the same viewpoint for both images, and making sure not to repeat any pairs of images that user has already seen. It is believed that approximately 30 visualization practitioners participated in the survey. (If a participant closed their web browser and restarted the survey, then they would appear as a new participant, making an exact count difficult.)

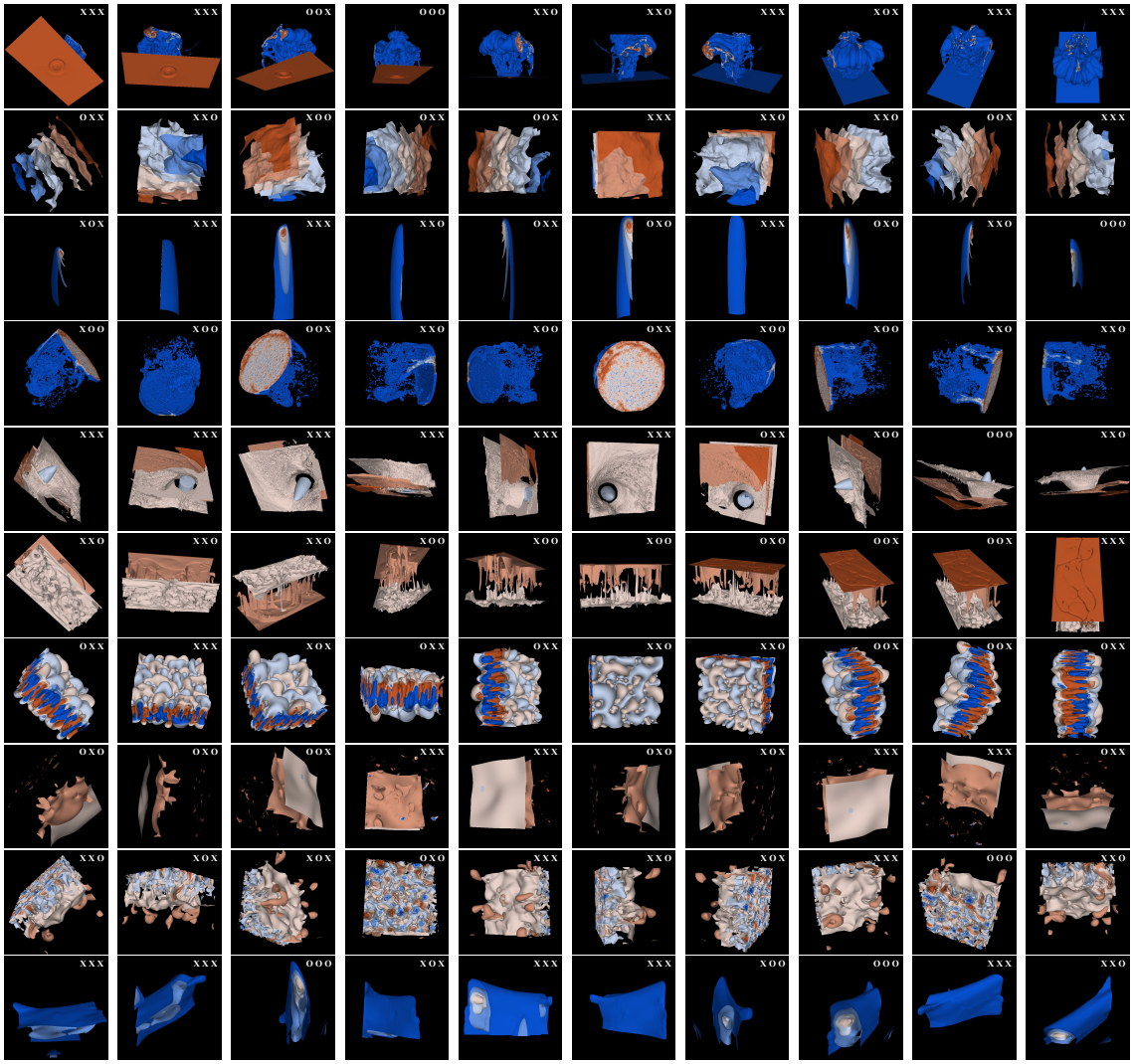


Figure 20. The image set for our corpus. Each data set was transformed into a multi-level isosurface, using six isovalues unique to each data set. Then, using the Fibonacci Lattice, each data set was rendered from ten equally-spaced camera positions around the data set. Each row of images is from the same scientific data set. Each column of images is using the same camera placement. The data sets are, from top to bottom: Asteroid, Constit, ExaAm Truchas, Fluid Dynamics, Hurricane, Mantle, Miranda, S3D-N2, S3D-UVEL, ExaSky Nyx. In the top right corner of each image is an annotation representing the Data Entropy score, Depth Entropy score, and Shading Entropy score, respectively, for each viewpoint. For each data set and metric, the scores were normalized to be between [0.09]. An **O** means the viewpoint has a metric score in the top 20% among the ten images. An **X** means the viewpoint has a metric score in the bottom 80% among the ten images.

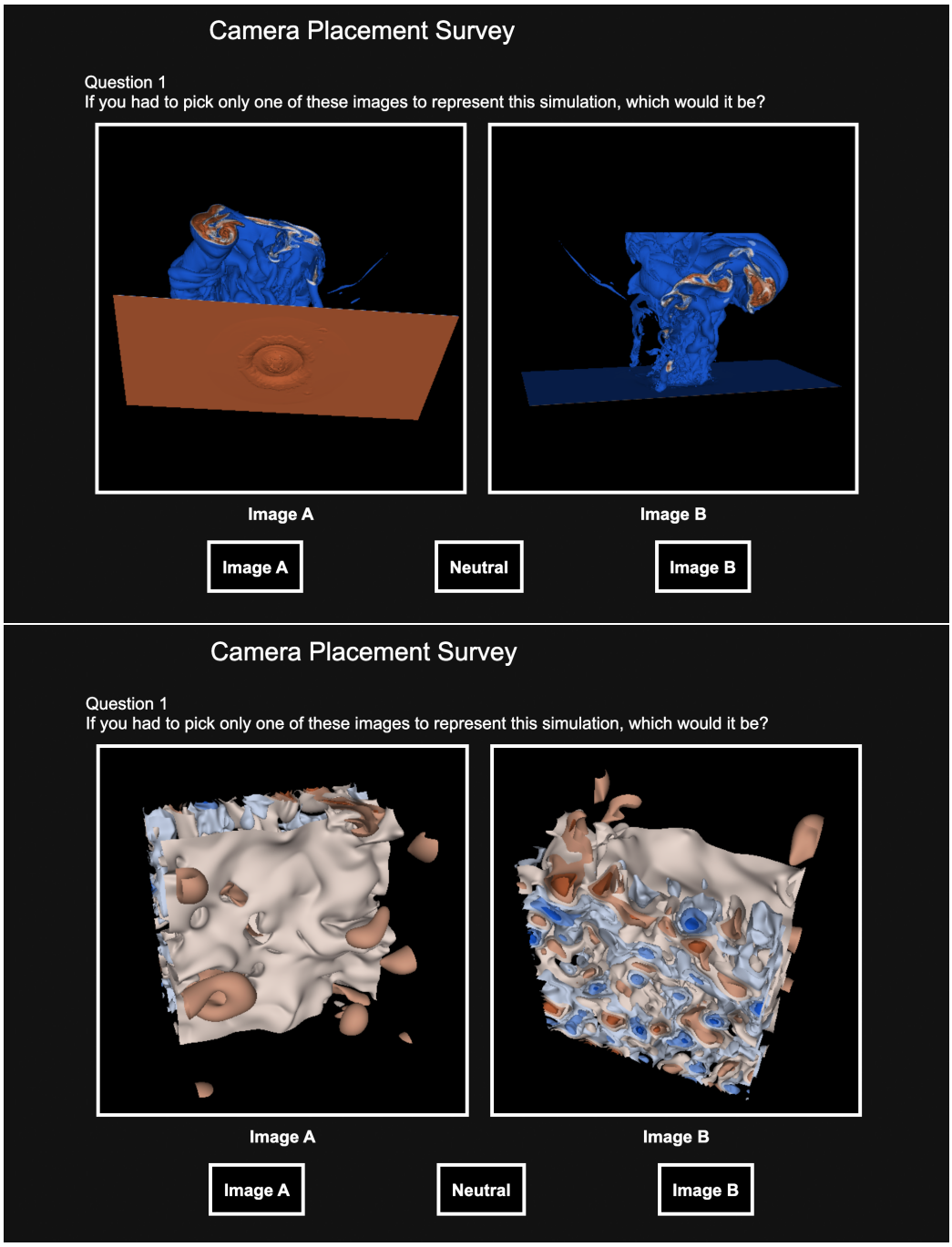


Figure 21. Examples of two questions from the user survey. The left image depicts a question between two viewpoints from the Asteroid data set. The right image depicts a question between two viewpoints from the S3D-UVEL data set. Users can select the image they feel is most representative of the simulation, or they can select neutral if they have no preference.

The survey resulted in 1266 responses, although 170 of these responses indicated the participant had no preference for one image over another. We discarded these "neutral" responses, resulting in 1096 entries where the participant had a preference. Each of these 1096 entries is a tuple of the form (D, C_i, C_j, R) where D is one of the ten data sets, C_i and C_j are camera positions, and R is the participant preference (i.e., $R = C_i$ or $R = C_j$).

With respect to metric information, we calculated the ten VQ metric values for each of the 100 combinations of camera position and data set, and these 1000 values complemented our corpus. The values were calculated using Ascent [71], which has implementations to calculate all ten VQ metrics (although some implementations have not yet been merged to the main Ascent repository). For the entropy calculations, data needs to be placed in a histogram with a fixed number of bins. The number of bins can change the distribution of the histogram and thus the entropy. For data entropy, we chose six bins, since there were six distinct scalar values (one for each isosurface). For shading entropy and depth entropy, we chose the common default of 100. Finally, Ascent was also used to generate the isosurfaces and imagery for the user study.

3.4 Evaluation Approach

As discussed previously in Section 3.2.1, our method uses camera metrics to construct oracles that attempt to predict human behavior. Our evaluation for a given oracle measures the extent that oracle can successfully predict participant preferences in our data corpus. For each entry (D, C_i, C_j, R) in the corpus, our evaluation approach provides the oracle with the corresponding VQ metrics and records whether the oracle predicted the user would prefer C_i or C_j. If the oracle correctly predicts R, then it receives one point. If not, then it receives zero points.

The oracle's score is the sum of these points over all 1096 entries, and the oracle with the highest score is the best, since it has made the highest number of correct predictions. Table 3 shows a notional example of this process.

Table 3. An example of evaluating an oracle on a notional corpus with four entries. Entries 1, 3, and 4 correspond to data set 1, while entry 2 corresponds to data set 2. This oracle would receive a total of two points, since it correctly predicts participant preferences for the first two entries, but is incorrect for the last two entries. Further, note that entries 1 and 4 involves the same combination of data set and cameras to compare, but the participants had different preferences, which is a situation that occurs in practice. This means that no oracle can achieve a perfect score for this corpus — the maximum score is three, since any oracle must make a poor prediction of user behavior for either entry 1 or entry 4.

Entry	Data Set	C_i	C_j	Participant Preference	Oracle Prediction
1	DS1	3	6	3	3
2	DS2	8	8	8	8
3	DS1	3	9	9	3
4	DS1	3	6	6	3

Despite having 1096 entries in our corpus, the maximum possible score is not 1096, because users sometimes disagree on which view is preferred (like entries 1 and 4 in Table 3). Therefore, none of our camera metric-based oracles should expect to get a score of 1096. We studied the corpus and determined the maximum achievable score is 952, i.e., when participants disagreed, the sum of the dissenting choices was 144.

3.5 Results

This section evaluates how well camera metric-based oracles can predict user preference. It is organized into three sections:

- Section 3.5.1 evaluates single-metric oracles, i.e., metrics that make predictions using only one type of camera metric.

Table 4. Correct predictions for each single-metric oracle. Percent correct is reported for both with respect to the maximum possible for our corpus (952 — see discussion in Section 3.4), and to the total number of entries (1096).

Metric	Correct Predictions	% (/952)	% (/1096)
Data Entropy	676	71.0%	61.6%
Shading Entropy	662	69.5%	60.4%
Maximum Depth	571	60.0%	52.1%
Depth Entropy	566	59.4%	51.6%
# of Visible Triangles	503	52.8%	45.9%
Visibility Ratio	502	52.7%	45.8%
Plemenos & Benayada	498	52.3%	45.4%
Viewpoint Entropy	492	51.6%	44.9%
VKL Distance	484	50.8%	44.2%
Projected Area	465	48.8%	42.4%

– Section 3.5.2 evaluates multi-metric oracles, i.e., metrics that make predictions using more than one type of camera metric.

– Section 3.5.3 considers the conditions where our top oracles can predict user behavior and where they cannot.

3.5.1 Single-Metric Oracles. The evaluation for each metric can be found in Table 4. The rate of correct prediction is surprisingly low for all metrics. An oracle that made random choices would be correct 50% of the time, and yet six of the metrics were unable to achieve this threshold. Certainly, these metrics (on their own) do not appear to be useful for the data sets in our corpus. The top performing metrics do include our new entropy-based metrics, although their rate of successful prediction is somewhat low. The best-performing metric, data entropy, is correct at slightly more than a 3-to-2 rate, although it does achieve 71% of the performance of a perfect oracle.

3.5.2 Multi-Metric Oracles. This multi-metric analysis begins by considering oracles that incorporate two metrics, with the top results listed

Table 5. This table displays results for two-metric oracles all three combination methods (**TIER**, **NORM**, **NONE**). There are 45 two-metric oracles, but only the best ten are shown for each combination method. The **TIER** method is prone to ties, and these ties were discarded, meaning the number of entries evaluated for **TIER** is lower than 1096. For example, the sum of the tiers for depth entropy and data entropy were equal for the camera pairs for 158 of the 1096 corpus entries, and so only the remaining 938 entries were considered. Further, its percentage is based on this lower number, i.e., 63.4% $\times$ 938 means this oracle made the correct number of predictions 595 times. The number of entries considered for **TIER** evaluations ranged over the 45 combinations from 842 to 1016. For **NORM** and **NONE**, the percentages reflect all 1096 entries.

	NORM			TIER			NONE		
Rank	Metric 1	Metric 2	%	Metric 1	Metric 2	%	Metric 1	Metric 2	%
1	Data Ent.	Shading Ent.	64.4%	Data Ent.	Depth Ent.	63.4%	Data Ent.	Shading Ent.	65.2%
2	Data Ent.	Depth Ent.	64.1%	Data Ent.	Max Depth	63.4%	Shading Ent.	Depth Ent.	64.8%
3	Shading Ent.	Depth Ent.	62.2%	Data Ent.	Shading Ent.	62.8%	Data Ent.	Depth Ent.	63.0%
4	Data Ent.	Max Depth	62.0%	Shading Ent.	Depth Ent.	62.2%	Shading Ent.	PB	60.5%
5	Data Ent.	Visible △'s	59.4%	Data Ent.	Visible △'s	59.5%	Data Ent.	Vis. Ratio	60.1%
6	Shading Ent.	Visible △'s	57.0%	Data Ent.	Vis. Ratio	59.0%	Shading Ent.	Vis. Ratio	60.0%
7	Shading Ent.	Vis. Ratio	56.0%	Data Ent.	VKL	56.8%	Data Ent.	Max Depth	59.5%
8	Depth Ent.	Max Depth	56.7%	Shading Ent.	Max Depth	56.6%	Data Ent.	PB	59.3%
9	Data Ent.	Vis. Ratio	56.5%	Data Ent.	Viewpoint Ent.	56.3%	Shading Ent.	VKL	57.2%
10	Shading Ent.	Max Depth	55.7%	Depth Ent.	Max Depth	56.1%	Depth Ent.	Max Depth	55.9%

in Table 5. This table demonstrates two important findings: (1) that our new entropy-based metrics are performing well as oracles and (2) that the method for combining metrics (i.e., **TIER**, **NORM**, or **NONE**) is not crucial. With respect to performance, each of the top oracles uses at least one of our entropy-based metrics. Further, the three combinations that involve two of our metrics (i.e., data entropy + shading entropy, data entropy + depth entropy, and shading entropy + depth entropy) rank as the top three for **NORM** and **NONE** and three of the top four for **TIER**. In fact, the top performer that does not have one of our entropy-based metrics are the 14^{th} best performers for **NORM** and **TIER** (visible triangles+maximum depth for both cases) and the 13^{th} best performer for **NONE** (VKL+maximum depth). With respect to combining metrics, while there is variation in the order and percentages, the overall trends are quite close: each

Table 6. This table displays results for three-metric oracles for all three combination methods (**TIER**, **NORM**, **NONE**). There are 120 three-metric oracles, but only the best ten are shown for each combination method. As discussed in Table 5's caption, the **TIER** method is prone to ties, and these ties were discarded. The number of entries considered for **TIER** evaluations ranged over the 120 combinations from 928 to 1051. Once again, the percentages for **NORM** and **NONE** reflect all 1096 entries. Finally, for formatting reasons, data entropy, depth entropy, and shading entropy are abbreviated DaE, DeE and ShE.

	NORM		TIER		NONE	
Rank	Metrics	%	Metrics	%	Metrics	%
1	DaE + ShE + DeE	65.7%	DaE + ShE + DeE	64.5%	DaE + ShE + DeE	68.0%
2	DaE + ShE + Max. Depth	64.1%	DaE + ShE + Vis. Ratio	63.3%	DaE + ShE + Vis. Ratio	65.0%
3	DaE + ShE + Vis. △'s	63.3%	DaE + ShE + Max. Depth	62.6%	DaE + ShE + PB	64.8%
4	DaE + ShE + Vis. Ratio	63.1%	DaE + ShE + VKL	61.8%	ShE + DeE + Vis. Ratio	64.5%
5	DaE + ShE + PB	61.4%	DaE + ShE + Visible △'s	61.7%	ShE + DeE + PB	63.8%
6	DaE + ShE + VKL	61.2%	DaE + DeE + Max. Depth	59.7%	DaE + ShE + VKL	62.9%
7	DaE + DeE + Max. Depth	60.7%	DaE + ShE + Viewpoint Entropy	59.6%	DaE + DeE + Vis. Ratio	61.8%
8	DaE + ShE + Projected Area	60.5%	ShE + DeE + Max. Depth	59.2%	DaE + DeE + PB	61.2%
9	DaE + ShE + Viewpoint Ent.	60.1%	DaE + DeE + Visible △'s	59.1%	ShE + DeE + Viewpoint Ent.	60.8%
10	ShE + DeE + Max. Depth	59.5%	DaE + DeE + Vis. Ratio	59.0%	ShE + Vis. Ratio + PB	60.5%

of the top combinations are in the 63%-65% range and many of the same pairs of metrics are repeated across the table.

Table 6 continues the analysis with oracles that incorporate three metrics. This table shows highly similar results to the two-metric analysis: our entropy metrics are good performers and the method for combining metrics (**TIER**, **NORM**, **NONE**) is not all that significant. Once again, each of the top ten performers involves at least one of our entropy-based metrics, and most involve two. Further, the top combination across all three combination methods is data entropy + shading entropy + depth entropy. One difference between the combination methods is that the **NONE** version yields the highest prediction rate (746 correct predictions with a maximum possible of 952). In terms of how the comparator (non-entropy) metrics performed, the top oracles were:

- **NORM** and **TIER** both had visible triangles + visibility ratio + maximum depth ranked as the 46^{th} best combination (out of 120), with successful predictions 50.4% and 50.1% of the time, respectively.

– **NONE** had VKL + visibility ratio + maximum depth ranked as its 35^{th} best combination (again out of 120), with successful predictions 54.3% of the time.

We repeated this analysis with four metrics, and found that 4-metric oracles were generally not as effective as the 3-metric oracles. One exception was the combination of data entropy, shading entropy, depth entropy, and PB (Plemenos & Benayada), which had a 68.4% winning percentage with **NONE**. That said, we are skeptical about the strength of this finding. On the one hand, PB does provide new insights that the entropy-based metrics do not have — it maximizes visible triangles and projected area per pixel, typically choosing viewpoints that "fill" the final image. On the other hand, the data in our corpus is noisy and the amount of improvement is small. Further, we are concerned that we run the risk of "reverse engineering" a result based on our corpus. In all, we conclude from this analysis that our entropy-based metrics do provide better prediction of user preference than previous methods, and also that the combination method is not important. We feel this conclusion is supported by the high rate that entropy-based metrics appear as top performers and in the invariance of the result across combination method.

3.5.3 Efficacy of Top Oracle. This section investigates how the top oracle (data entropy + shading entropy + depth entropy, combined with **NONE**) performed on the corpus.

Table 7 presents analysis about how the oracle performs for different types of cameras. Specifically, each camera is classified as "**POOR**," "**FAIR**," "**GOOD**," or "**VERY GOOD**" and the analysis considers how the oracle performs for each of the combinations (e.g., when the oracle is asked to choose between a "**POOR**" and "**GOOD**" camera). We classify the cameras based on their win percentage, i.e., the rate that an individual image was preferred by

Table 7. Performance statistics for our top oracle. As an example of how to interpret this table, the data from the **GOOD/POOR** entry means that there were 222 instances in our corpus where our oracle was asked to choose between one camera that was **GOOD** and one camera that was **POOR**, and it correctly matched participant preference 188 times, which was an 85% success rate. The Sum column provides statistics about behavior for one camera grouping. For example, there were 470 instances in our corpus where at least one of the cameras was **POOR**, for which our oracle correctly matched participant preference 82% of the time. The Sum column involves double counting some of the corpus entries, e.g., entries with both **GOOD** and **FAIR** are counted in both the **GOOD** and **FAIR** sums. Some table cells have higher counts because the number of cameras for each type varies; there are 34 **GOOD** cameras and 22 of the other three types. Finally, table cells are colored by their performance: a success rate of 70% or more is colored green, 60%-70% is colored yellow, and less than 60% is colored pink.

	POOR	FAIR	GOOD	VERY GOOD	Sum
POOR	82% (23/28)	75% (55/73)	85% (188/222)	80% (118/147)	82% (384/470)
FAIR	75% (55/73)	60% (30/50)	54% (88/162)	61% (69/114)	61% (242/399)
GOOD	85% (188/222)	54% (88/162)	58% (57/99)	60% (107/177)	67% (440/660)
VERY GOOD	80% (118/147)	61% (69/114)	60% (107/177)	46% (11/24)	66% (306/462)

participants. For example, camera position #6 for the Mantle data set was one of the most preferred images, being preferred by users in 28 out of 31 comparisons, i.e., a "win percentage" of 90.3%. We label as follows: 0%-25% as **POOR**, 25%-50% as **FAIR**, 50%-75% as **GOOD**, and 75%-100% as **VERY GOOD**. While our oracle does not have access to either win percentage or these labels, they are useful for postmortem analysis of oracle behavior. In terms of findings, our oracle appears to be most effective at predicting user preference when **POOR** cameras are involved, with an 82% efficacy, which was the highest of any group by a large margin. When a participant is asked to choose between two cameras that are **GOOD** or **VERY GOOD**, our oracle is only 58% effective (175/300). Possibly these images are both adequate to the participant, and so other factors, such as esthetics, become more important. Overall, this table shows that we perform

relatively similarly for all types of cameras. Moreover, it does not show evidence that our oracle is under-performing for certain types of comparisons.

Table 8. The rate of successful predictions by our top oracle per data set.

Data Set	Prediction Rate
Asteroid	74.2%
Constit	52.2%
ExaAM	76.5%
Fluid Dynamics	80.8%
Hurricane	60.3%
Mantle	79.1%
Miranda	52.5%
S3D-N2	57.7%
S3D-UVEL	72.7%
ExaSky Nyx	71.5%

Table 8 presents prediction rate by data set. Three of the data sets — Constit, Miranda, and S3D-N2 — have success rates below 60%, and a fourth, Hurricane, is just above 60%. The remainder of the data sets are at or above the oracle average. Visual inspection of the images for these four data sets shows:

- Constit has many views that are similar. Participants preferred those with empty space between the isosurfaces, presumably the other views created confusion with occlusion issues. None of the entropy-based metrics would assist with this issue.

- Hurricane is composed of several layers with high rates of occlusion, undoubtedly causing issues for both data entropy and depth entropy. Additionally, this data set is easily identifiable and an uncommon choice of esthetics may influence user preference.

- Miranda has blue/orange surfaces at the boundary of the volume that are (in the opinion of the authors) not as interesting as the isosurfaces at the mixing

layer. That said, data entropy rewards showing more of these blue/orange surfaces. When we modified our oracle to ignore data entropy, the prediction rate went to 69.7%, in line with the oracle average.

– S3D-N2 has small features that are clearly visible from some images, but not from others. Users preferred the images with these features, but the data entropy calculation did not reflect their presence since its calculations were dominated by the other surfaces.

All of these observations suggest possible future improvements for an oracle-based scheme.

3.6 Conclusion

The research premise of this study is that the entropy-based metrics are good predictors, especially in combination, and we feel our results provide strong evidence to support this premise. We do believe that we could look further at ways to combine metrics (such as weighted combinations) and decision-tree type oracles ("if the data entropy is better then choose this camera, else look at shading entropy...") and optimize those approaches for this given corpus. But such optimizations would have to be verified on a different corpus with different data set and users; we view this as future work. We also believe the shortcomings identified with Constit, Hurricane, Miranda, and S3D-N2 suggest future avenues of improvement.

We (surprisingly) recommend the **NONE** combination method of adding the three scores together. While such an approach creates apples-to-oranges concerns when involving non-entropy metrics, it is appropriate when adding three entropy metrics together. Our motivation for this recommendation is not because **NONE** got the highest prediction percentage, but because it requires no additional

knowledge. **NORM** and **TIER** require evaluating multiple cameras to establish a baseline for what should go in a "good" or "bad" tier or what should be normalized to "1" or "10." **NONE** does not require these extra calculations, which is an advantage in an in situ setting.

There are two major areas of future work: (1) how the study itself can be improved and extended, and (2) how the results of this study can be used for automating in situ camera placement. With respect to improving the study, we would like to continue to expand our corpus of data and evaluate additional metrics and oracles. In particular, our corpus consisted of multi-level isosurface imagery, and other visualizations may have differing results. Further, we limited our consideration of combined metrics due to concerns of overfitting, but an additional corpus would enable us to optimize combinations on one corpus and validate on the other. Other extensions could include enhancing the survey, for example to include stereo imagery, or enhancing the overall approach, for example considering variation in isolevels as well as camera position. With respect to in situ automation, this research direction is tackled in the remaining chapters.

CHAPTER IV

AUTOMATIC IN SITU CAMERA PLACEMENT FOR LARGE-SCALE SCIENTIFIC SIMULATIONS

This chapter is co-authored work that is in submission. Contributions to this work are the following: I provided initial parallelized implementation of the VQ metrics, as well as executing the performance and search studies. I was also the primary author of this work. Manish Mathai re-parallelized the implemented VQ metrics using VTK-m and helped in the editing of the manuscript. Stefan Fields explored search algorithms in the early stages of this work. Hank Childs provided extensive feedback over the course of this work and helped in editing the manuscript.

4.1 Introduction

There are two major challenges to deploying VQ metrics for in situ camera placement. The first challenge is selecting VQ metrics that will fit within an in situ environment, i.e., they can be enhanced to run in a distributed-memory environment and can operate quickly on many-core architectures. The second challenge is finding a quality viewpoint quickly. While there are an infinite number of possible camera positions, it is critical to find a good camera placement without having to consider 100s or even 1000s or viewpoints.

The main contribution of this chapter is to address these two challenges:

- We develop a parallelization scheme for two common patterns of VQ metrics that enable VQ metrics to be deployed in a parallel (distributed-memory+shared-memory), in situ setting. This is the first-ever approach for parallelizing calculation of these metrics. We also establish that the approach is performant.

– We develop search algorithms that trade off between camera quality and total execution time. We also evaluate these algorithms, and provide practical suggestions about "sweet spots" in these tradeoffs.

In all, this work provides a pragmatic and useful approach for in situ automation of camera placements. It also shows that automatic camera placement can be performed efficiently in situ.

In our study, we incorporate the same eleven VQ metrics Marsaglia et al. [92] evaluated in their user preference study. The selection of these VQ metrics is based on their potential for in situ implementation: small memory footprint, low communication overhead, and quick execution time. The eleven metrics are summarized in Table 9.

4.2 Our Method

This section describes our method in two parts: Section 4.2.1 describes our parallelization for the calculation of VQ metrics, while Section 4.2.2 describes how we search for a good camera placement. In terms of relationship between the two parts, the search algorithm operates by evaluating VQ metrics at potential camera positions, and it uses the parallelization approach to quickly calculate these metrics in a distributed-memory, in situ setting.

4.2.1 VQ Metric Parallelization.

The eleven VQ metrics we consider all operate by analyzing the result of a rendering process. That is, a rendering process takes a data set as input and produces an image, and then a VQ metric analyzes the image to produce a number. However, each VQ metric alters the rendering process to accomplish its goals. For example, the "visible triangles" metric annotates each triangle with a unique identifier prior to rendering, and the result of the rendering process is an image where pixels contain triangle

Table 9. Descriptions of the 11 VQ metrics used in this research.

VQ Metric	Definition
Data Entropy	Maximizes the entropy of the visible field data from some viewpoint [92].
DDS Entropy	The sum of the three VQ metric scores: Data Entropy + Depth Entropy + Shading Entropy [92]. The Marsaglia user study showed this metric was most aligned with user preference.
Depth Entropy	Maximizes the entropy of the visible depth data from some viewpoint [92].
Max Depth	Maximizes the visible depth from some viewpoint [118].
Plemenos and Benayada (PB)	Maximizes the number of visible triangles as well as the resolution of the rendered image [102].
Projected Area	Maximizes the visible projected area of the data set from some viewpoint [102].
Shading Entropy	Maximizes the entropy of the visible shading coefficients from some viewpoint [92].
Viewpoint Entropy	Maximizes the entropy of the projected area from some viewpoint [134].
Visibility Ratio	Maximizes the ratio of visible surface area over the total surface area [102].
Visible Triangles	Maximizes the total number of visible triangles from some viewpoint [102].
Viewpoint Kullback-Leibler distance (VKL)	Minimizes the distance between the normalized distribution of projected areas in Image Space and the normalized distribution of projected areas in World Space [116].

identifiers instead of colors. For this metric, if triangle T is visible from a given camera position at pixel P, then the image data for pixel P will contain T. The metric can then count the number of visible triangles by counting the number of unique identifiers in the image.

We implemented our algorithm in the Ascent in situ library [71], which makes use of VTK-m [95]. VTK-m operates on shared-memory architectures, and Ascent adds distributed-memory parallelism via MPI. VTK-m employs a portable parallelism approach: algorithms are implemented using data parallel

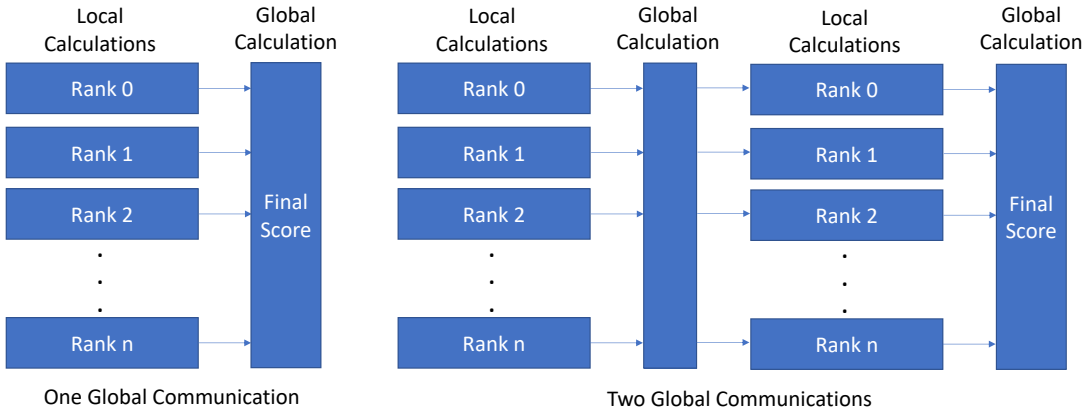


Figure 22. As input, each metric receives the simulation data, including both field data and geometry, as well as a rendering of the visible field data and geometry from some viewpoint. Using this input, the implemented metrics require either one or two global coordinations when computing a viewpoint score.

primitives, which then get executed efficiently on devices using an appropriate back end (CUDA, OpenMP, Kokkos). VTK-m and Ascent already had parallel rendering capability, which could utilize shared-memory and distributed-memory parallelism, through VTK-m and Ascent respectively. Further, the approach for parallel rendering is well established in the visualization community: each MPI task renders its own data to make a "sub-image" and then all MPI tasks participate in a "compositing" phase where the "sub-images" are combined to make a single image. For our approach, we adapted Ascent's rendering to have the requisite information to carry out a VQ metric calculation (e.g., triangle identifiers), had Ascent perform a render, and then analyzed the result in our VQ routines. The VQ calculations were implemented using a combination of VTK-m (using shared-memory parallelism to analyze images to calculate metrics) and MPI (to coordinate calculations across MPI tasks).

At a high level, the VQ metrics all follow one of two execution patterns, as shown in Figure 22. Depending on the metric, the calculations may require

one or two global communications. In the former case, the metrics receive the data, perform local calculations, and then globally coordinate the final metric score. In the latter case, the individual ranks first perform any local calculations, second they coordinate globally, third they again individually perform any local calculations, and then finally they do one more global coordination to calculate the final score.

Data Entropy is an example of a VQ metric that requires only one global communication. From Ascent, this metric receives the minimum and maximum field values, as well as a rendering of the visible field data from some camera placement. In our infrastructure, rank 0 is the only process that receives the composited render of the visible data. From here, rank 0 sorts the visible data present in each pixel into a histogram comprised of 1000 bins that are equally spaced using the field range. After creating a histogram, rank 0 can calculate the probability of each field value and calculate entropy using Shannon's Entropy. Lastly, rank 0 broadcasts the final entropy score to the other ranks.

Visibility Ratio is an example of a VQ metric that requires two global communications. From Ascent, this metric receives a rendering that details the visible triangles via triangle identifiers. (Again, rank 0 is the only process that receives the composited render of the visible triangles.) In the first phase of local work, each rank calculates the surface area of their local geometry. The first global communication is an MPI_Reduce summation that adds up each rank's local surface area and returns a global surface area to rank 0 (the root process). In the second phase of local work, rank 0 calculates the visible surface area from the rendering of visible triangles and then calculates the visibility ratio by dividing

the visible surface area by the global surface area. Lastly, rank 0 broadcasts the final visibility ratio to the other ranks.

4.2.2 Viewpoint Search Algorithm. An important consideration for this algorithm is the "stability" of VQ metrics as the camera moves. If small changes in camera position consistently lead to large changes in metrics scores, then it would be difficult to search the space without performing many evaluations. On the other hand, if scores vary somewhat smoothly, then it is easier to search with few evaluations. Section 4.3.2.1 explores this topic in detail. That said, the findings from that section provide intuition behind our search algorithms: scores vary somewhat smoothly as the camera position changes.

Much like the approach used in the popular Cinema project [3], our search algorithms consider camera locations on the surface of a sphere that bounds the data set. Using Spherical Coordinates, we divide the surface into equal segments, ϕ_m and θ_n, for some integers m and n, that creates a grid search-space of size $m * n$. A user can then decide a camera budget that dictates how many camera placements to evaluate with each search algorithm.

We developed four search algorithms. That said, one of the algorithms has three variants, meaning we have six algorithms overall (three algorithms with no variants, and one algorithm with three variants). Given a search space and camera budget, the algorithms operate as follows:

- **Random Search**: each new camera position is placed at a random location, as shown in Algorithm 1.
- **Diagonal Search**: this approach travels along a diagonal of the search space, and allows for user-specified spacing between the considered camera placements, as shown in Algorithm 2. The variants are due to skipping

behavior defined by the sample rate, which greatly affects performance. In order to hit every grid point and evaluate the respective camera, it is critical that the chosen sample rate is co-prime to the dimensions of each axis. That is, the chosen sample rate needs to be co-prime with both n and m, meaning the only common divisor is one. The sample rates selected for evaluation in Section 4.3.2.2 are 7, 23, and 43.

- **Space-Filling Curve Search**: as more and more potential camera positions are considered, they are placed using the Morton (Z-order) space-filling curve approach, as shown in Algorithm 3. This ensures that each additional camera position is placed into the largest unexplored region of possible camera positions.

- **Linear Search**: iterate over the camera positions one-by-one, with each camera position adjacent to the previous one, as shown in Algorithm 4. This algorithm does a poor job of sampling the space and is intended only as a reference.

Algorithm 1 Random Search

```
max_score= −FLT_MAX
count = 0
while count < camera_budget do
  cam_pos = Math.Random() ∗ n ∗ m
  if score(cam_pos) > max_score then
    max_score = score(cam_pos)
    count++
  end if
end while
```

Algorithm 2 Diagonal Search with sample rate

```
max_score= −FLT_MAX
count = 0
while count < camera_budget do
  phi_pos = count ∗ sample_rate % m
  round = (int)(count ∗ sample_rate)/m % m
  theta_pos = (round + phi_pos) % n
  cam_pos = GetCamera(phi_pos,theta_pos)
  if score(cam_pos) > max_score then
    max_score = score(cam_pos)
    count++
  end if
end while
```

Algorithm 3 Space-Filling Curve Search

```
max_score= −FLT_MAX
count = 0
while count < camera_budget do
  cam_pos = morton_space_filling_curve(count)
  if score(cam_pos) > max_score then
    max_score = score(cam_pos)
    count++
  end if
end while
```

Algorithm 4 Linear Search

```
max_score= −FLT_MAX
count = 0
while count < camera_budget do
  phi_pos = count % m
  theta_pos = (int)(count)/m % n
  cam_pos = GetCamera(phi_pos, theta_pos)
  if score(cam_pos) > max_score then
    max_score = score(cam_pos)
    count++
  end if
end while
```

4.3 Results

Our results are organized into three phases. Phase 1 focuses on the performance of individual Viewpoint Quality metrics in a parallel setting. Phase 2 focuses on the efficacy of our search algorithms to find an optimal image quickly. Finally, Phase 3 considers holistic behavior by running our algorithm in an in situ setting.

4.3.1 Phase 1: Parallel Performance of Individual Metrics. To evaluate parallel performance, we ran the Lulesh proxy application on NERSC's Cori supercomputer. Our runs used 27 nodes with one MPI task per node. Each MPI task incorporated shared-memory parallelism through OpenMP, and had access to 32 threads. The Lulesh simulation was of a Sedov blast problem, which we ran for 100 cycles and evaluated 10 camera positions per cycle. Each MPI task had a data size of 342^3, for a total data size of 1020^3.

Table 10 contains the results of our experiments. The major result is that the rendering stage dominates the majority of the computation, and our separate phases for metric evaluation add minimal overhead. In other words, our method takes about long as rendering a single image. Of course, our overall approach involves evaluating many camera positions. Phase 2 will explore how many camera positions are necessary. Phase 3 will then add more context to these parallel performance results, since it will consider overall in situ encumbrance.

4.3.2 Phase 2: Evaluating Viewpoint Search Algorithms. The second contribution of this work evaluates viewpoint search algorithms on scientific data sets. For this contribution, we first examine the stability of the search space. We then evaluate how quickly search algorithms can find a quality viewpoint.

Metric	Pre-Processing*	Rendering	Local Work 1	Global Comm 1	Local Work 2	Global Comm 2	Total
Data Entropy	7.80E-05	1.01E-02	1.01E-04	1.08E-05	-	-	1.03E-02
Depth Entropy	7.80E-05	1.01E-02	9.59E-05	8.09E-06	-	-	1.03E-02
Max Depth	7.80E-05	1.01E-02	1.59E-05	1.00E-05	-	-	1.02E-02
PB	7.80E-05	1.01E-02	1.02E-03	1.01E-05	-	-	1.12E-02
Projected Area	7.80E-05	1.01E-02	1.00E-03	9.34E-06	-	-	1.12E-02
Shading Entropy	7.80E-05	1.01E-02	9.91E-04	9.44E-06	-	-	1.12E-02
Viewpoint Entropy	7.80E-05	1.01E-02	6.81E-06	7.34E-06	1.00E-03	1.08E-05	1.12E-02
Visibility Ratio	7.80E-05	1.01E-02	2.16E-08	9.84E-06	1.00E-03	1.06E-05	1.12E-02
Visible Triangles	7.80E-05	1.01E-02	1.01E-03	1.21E-05	-	-	1.12E-02
VKL	7.80E-05	1.01E-02	5.05E-06	9.30E-06	1.00E-03	8.12E-06	1.12E-02

Table 10. The total execution times, in seconds, for each metric to evaluate a single viewpoint. Note: Pre-processing is a stage in the workflow that only needs to be executed once per cycle, no matter how many camera placements are considered. The rendering stage, as well as the metric evaluations, need to be excuted for each considered camera placement.

4.3.2.1 Stability of Search Space.

The values in the search space depend on both the VQ metric and the data being rendered. For our evaluation, we considered the VQ metrics from Table 9. For data sets, we used the same ten data sets from the Marsaglia et al. [92] user study on camera placement for large data visualization. These data sets are:

- Asteroid: A data set of a deep water impact of an asteroid [100].
- Constit: A material sciences data set that probes the deformation response of polycrystalline materials [32].
- ExaAm Truchas: A materials science data set that looks at effects within micro-structures of Additive Manufacturing (AM) [16].
- ExaSky Nyx: A cosmological data set that looks at gas dynamics [5].
- Fluid Dynamics: A fluid dynamics data set that models a cylindrical flow of water [70].
- Hurricane: A weather data set of Hurricane Isabel [1].

– Mantle: An earth sciences data set that models the Earth's mantle [120].

– Miranda: A hydrodynamics data set of large-scale turbulence [39].

– S3D-N2: A combustion data set of field data N2 [132].

– S3D-UVEL: A combustion data set of field data U Velocity [132]

For each data set, we calculated each VQ metric for ten-thousand viewpoints. The viewpoints were constructed in the same way as the search algorithms in Section 4.2.2: along the surface of a sphere, and taking even increments in ϕ and θ in Spherical Coordinates ($\phi_m = \theta_n = 100$). The ten-thousand scores were then rendered as heatmaps (high scores are white, low scores are dark orange and black):

– Figure 23 shows heatmaps for two data sets, including thumbnails of the images with high or low scores using DDS Entropy,

– Figure 24 shows heatmaps for 10 VQ metrics for a single data set, and

– Figure 25 shows heatmaps for 10 data sets using the DDS Entropy metric (the metric that best predicts user preference from Marsaglia's user study).

While there are sharp cliffs as occlusions change, the "high" regions for each metric are (for the most part) large and easy to find. Further, the metric DDS Entropy, shown in Figure 25, has large "white" regions, meaning good viewpoints are not sparse nor randomly appearing. In their user study, Marsaglia found that users disagreed regularly about "good" views, but were consistent in rejecting "bad" views. Based on the DDS Entropy heatmaps in Figure 25, these bad views are relatively easy to identify and avoid, i.e., find a camera placement a white region

instead of an orange or black region. Whereas the score heatmaps in Figure 24 has several VQ metrics where the local optimums are tightly clustered and may be hard to locate quickly. Overall, we find these heatmaps to be promising with respect to the searchability of the space.

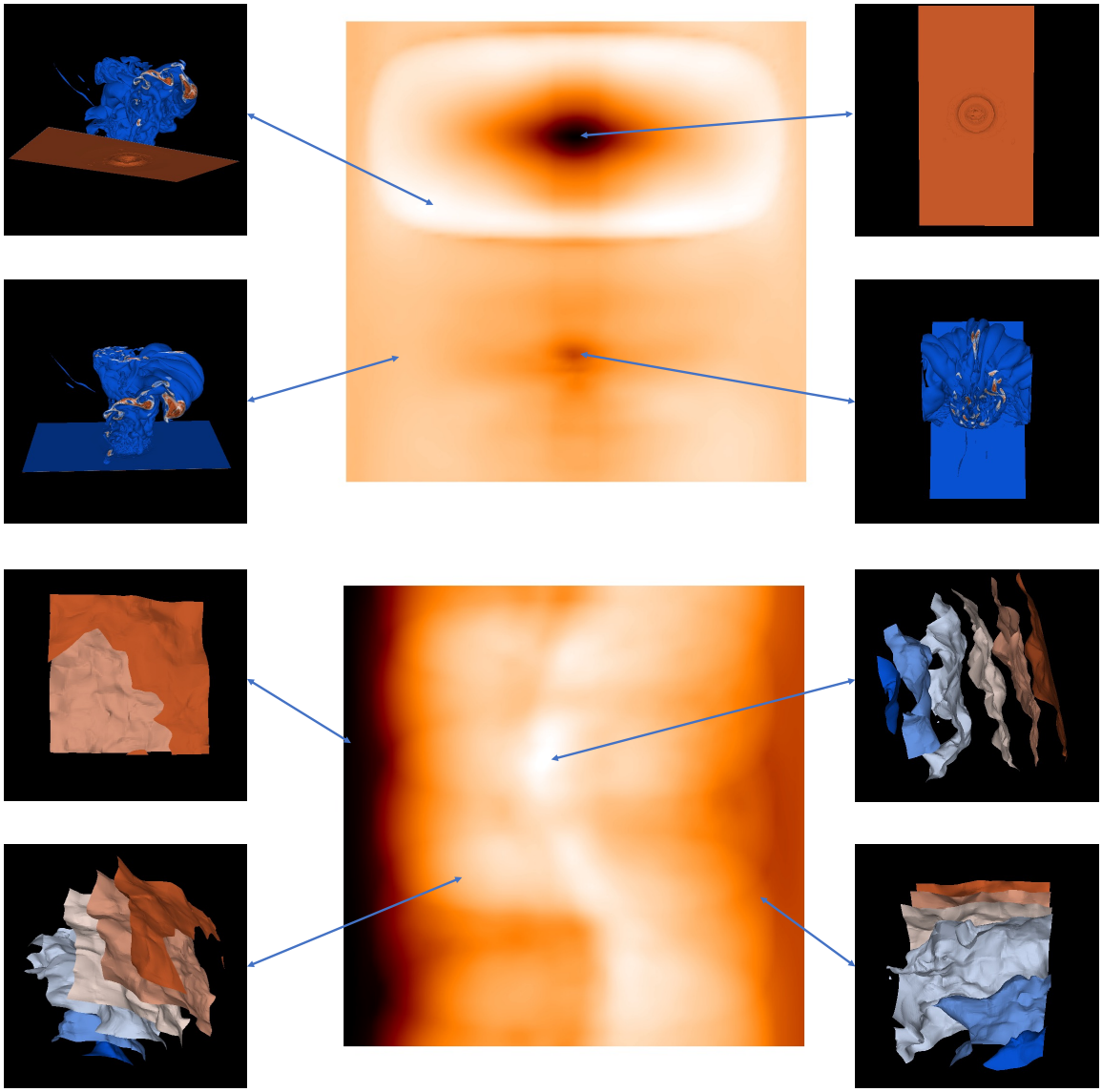


Figure 23. Annotated heatmaps of the Asteroid (top) and Constit (bottom) data sets. For the Asteroid data set, the DDS Entropy scores from best to worst are: top left, bottom left, bottom right, top right. For the Constit data set, the best to worst are: top right, bottom left, bottom right, top left.

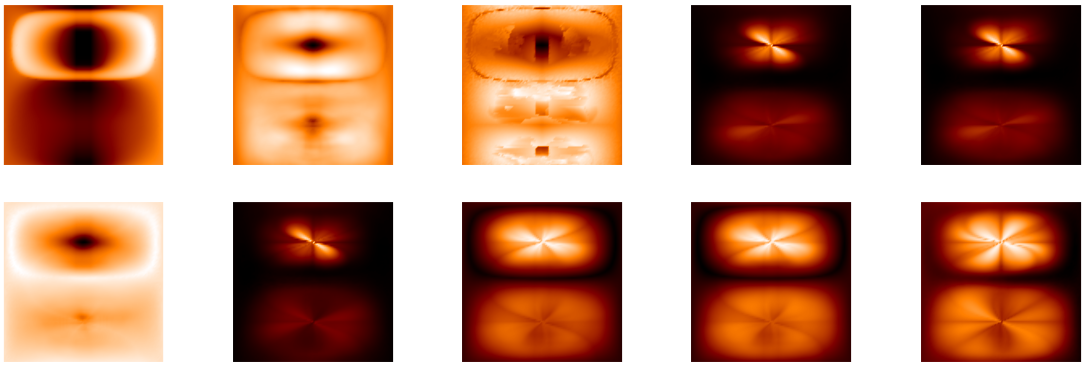


Figure 24. Ten heatmaps corresponding to ten VQ metrics applied to the Asteroid data set. The VQ metrics are, from top left to bottom right: Data Entropy, Depth Entropy, Max Depth, PB, Projected Area, Shading Entropy, Viewpoint Entropy, Visibility Ratio, Visible Triangles, VKL. The Asteroid heatmap for DDS Entropy can be found in Figure 25. These heatmaps show that the searchability of the space is dependent on VQ metric.

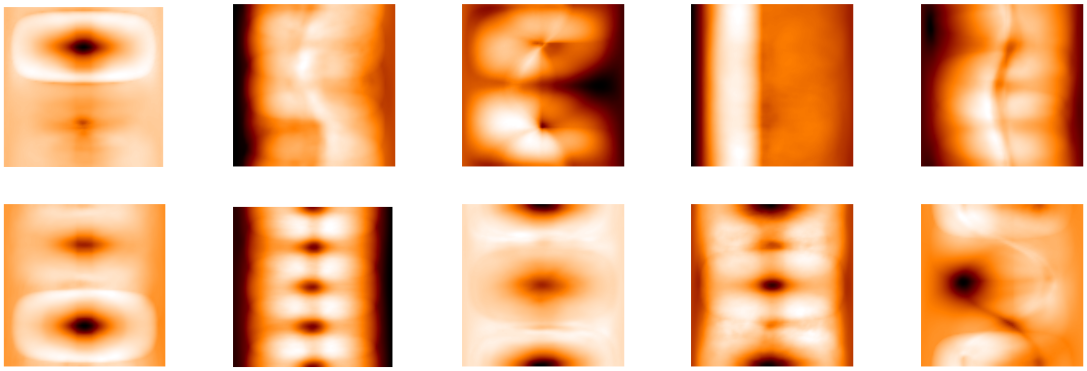


Figure 25. Heatmaps of DDS Entropy values as camera position changes for all ten data sets. The data sets are, from top left to bottom right: Asteroid, Constit, ExaAm Truchas, Fluid Dynamics, Hurricane, Mantle, Miranda, S3D-N2, S3D-UVEL, ExaSky Nyx. These heatmaps show that the higher scores or "hot spots" are not sparse nor randomly appearing.

4.3.2.2 Search Algorithm Evaluation.

This evaluation considered all algorithms from Section 4.2.2. We tested our search algorithms on the heatmaps from Section 4.3.2.1, evaluating how many camera positions each search algorithm needs to evaluate in order to find a quality viewpoint.

Search Algorithms	Percentile				
	80^{th}	85^{th}	90^{th}	95^{th}	99^{th}
Linear	1330.8	1592.4	1816	2514	4025.6
Random	25.1	31.8	83.3	210.2	921
Diagonal 7	91.7	126.8	145.5	322.9	1141.2
Diagonal 23	33.5	40	80.3	253.9	1051.2
Diagonal 43	23.3	30.3	69	138.6	799.1
Space-Filling Curve	29	36	83	147.9	712.2

Table 11. The average number of camera positions each search algorithm had to evaluate in order to find a camera position with a score in a given percentile. This average is taken over all VQ metrics.

Search Algorithms	Percentile				
	80^{th}	85^{th}	90^{th}	95^{th}	99^{th}
Linear	267.6	350.2	507.9	1428	3446.6
Random	2.2	3.2	6.8	22	398.9
Diagonal 7	7.2	12	20.8	60.5	359.1
Diagonal 23	3	4.6	7.9	44	307.6
Diagonal 43	2.1	3.8	5.4	19.2	104.6
Space-Filling Curve	3.9	7.6	11.7	28.2	204.9

Table 12. The average number of camera positions each search algorithm had to evaluate in order to find a camera position with a score in that percentile. This average is taken over just the Entropy VQ metrics that Marsaglia et al. found best predict user preference.

Table 11 shows the average number of camera positions it takes for each search algorithm to find a viewpoint that has a score in the $80^{th}, 85^{th}, 90^{th}, 95^{th}$, and 99^{th} percentiles for each VQ metric. The best performers were primarily the Diagonal search algorithms followed by Random Search and then Space-Filling Curve. Diagonal 43 performed the best, finding a viewpoint in the 80^{th} percentile after considering 23.3 camera placements on average, followed closely by Random Search and Space-Filling Curve, which had to consider 25 and 29 camera placements on average, respectively.

We repeated this analysis, but limited the VQ metrics considered to just the Entropy metrics that the Marsaglia et al. study showed most closely align with user preference [92]. The results are in Table 12. Again, Diagonal 43 performed the best, being able to find a viewpoint with a score in the top 80^{th} percentile after considering, on average, 2.1 viewpoints. Interestingly, one of the best performers is the Random Search. This is most likely due to the nature of the data sets and the fact that good viewpoints are not sparse. Overall, almost all of the search algorithms can find a viewpoint in the top 80^{th} and 85^{th} percentiles rather quickly. But, if a user is wanting a viewpoint in the 90^{th} percentile or higher, the search for a viewpoint could be substantially longer and the user will have to decide if performing 10X more searches is worth an image that is 10% better.

4.3.3 Phase 3: In Situ Evaluation. This phase builds on the previous two phases by performing an situ evaluation. Section 4.3.1 demonstrated that parallel VQ metric calculation can be executed quickly, adding little overhead to the more costly rendering process. Section 4.3.2 provided insight into best practices when searching for a camera placement. That said, this phase did its evaluation by using data that would be prohibitive to calculate in an in situ setting.

For this phase, we ran a scientific simulation with in situ camera placement search, and evaluated tradeoffs between in situ constraints and the quality of a VQ metric's chosen camera placement. Specifically, we again ran Lulesh on NERSC's Cori supercomputer, with the same level of parallelism as Phase 1. We executed Lulesh for 100 cycles, and for every cycle evaluated each metric using one of five different budgets: 5, 10, 20, 50 and 100 camera placements. The five budgets control how many camera placements are evaluated. The camera placements were chosen using Diagonal 43, i.e., the best performing search algorithm from Section 4.3.2.2, Note that our search algorithm starts from the same location for a given camera budget, meaning that the set of placements for a smaller budget are a subset of the set of placements for a larger budget. In terms of measurement, we timed the in situ execution of rendering the budgeted camera placements and calculating their respective VQ metric scores.

Table 13 shows the in situ execution results for a single cycle, as well as the maximum score achieved for each VQ metric for each camera budget. Among the eleven evaluated VQ metrics, only one metric, VKL, found its best camera (for this cycle) amongst the first 5 cameras considered, meaning this metric did not benefit when considering up to 95 more cameras. The ten other VQ metrics all benefited at least once to an increase in the number of cameras considered. Six of those ten VQ metrics experienced only one increase across all budgets: PB, Projected Area, Viewpoint Entropy, and Visible Triangles found their best camera placement when considering 10 cameras, whereas Data Entropy and Visibility Ratio found their best camera placement when considering 20 cameras. VQ metrics DDS Entropy, Depth Entropy, Max Depth, and Shading Entropy all experienced multiple increases across budgets. Max Depth found their best score when considering 50 cameras, while

DDS Entropy, Depth Entropy, and Shading Entropy found their best score when considering 100 cameras. Overall, for the given cycle, the majority of metrics found their best camera placement when considering 20 cameras or less, whereas only four metrics benefited from considering 50 or 100 cameras.

We then evaluated the average increase in score for each metric when increasing the camera budget over all 100 cycles of the Lulesh simulation. We did this by normalizing the VQ metric scores for each cycle using the scores from all budgets. Once normalized, we calculate the percentage change when increasing the camera budget, and then took the average across all cycles. Table 14 shows the average percentage increase in score for each metric when the budget was increased.

VQ Metric	5 Cameras		10 Cameras		20 Cameras		50 Cameras		100 Cameras	
	Max Score	Time (s)	Max Score	Time (s)	Max Score	Time (s)	Max Score	Time (s)	Max Score	Time (s)
Data Entropy	1.633	1.523	1.633	2.786	1.634	5.057	1.634	12.922	1.634	26.167
DDS Entropy	9.542	1.858	9.627	3.102	9.74	5.672	9.965	14.489	10.162	29.077
Depth Entropy	4.248	1.522	4.331	2.783	4.363	5.055	4.536	12.912	4.686	26.135
Max Depth	0.0124	1.513	0.01253	2.758	0.01257	5.007	0.0131	12.774	0.0131	25.899
PB	7.066	1.525	7.119	3.024	7.119	5.542	7.119	14.130	7.119	28.611
Projected Area	6284440	1.644	6332160	3.023	6332160	5.543	6332160	14.123	6332160	28.581
Shading Entropy	3.680	1.642	3.680	3.020	3.746	5.540	3.763	14.109	3.844	28.565
Viewpoint Entropy	3.872	1.654	3.876	3.027	3.876	5.544	3.876	14.151	3.876	28.638
Visibility Ratio	0.8660	1.652	0.8660	3.028	0.8692	5.545	0.8692	14.137	0.8692	28.268
Visible Triangles	147	1.643	148	3.024	148	5.542	148	14.125	148	28.597
VKL	-0.2632	1.655	-0.2632	3.027	-0.2632	5.551	-0.2632	14.148	-0.2632	28.625

Table 13. Metric scores and execution time for all five budgets and all VQ metrics for our Lulesh experiments using the Diagonal 43 search algorithm. Using the first row as an example, with a budget of five cameras, the best camera with respect to Data Entropy took 1.523s and produced a score of 1.633. If the budget was ten cameras, then the search cost went up (2.786s), but the score did not go up. Going up to a budget of twenty cameras, the score went up very slightly, but the time also got longer. This cell is colored pink to indicate its value increased with the larger budget. This table shows the results for cycle 50, which is representative.

Few of the metrics showed any substantial increase in metric score. Depth Entropy and Max Depth had the most substantial percentage increase, each experiencing an average increase in score of 11%, when increasing the considered cameras from 20 to 50. Depth Entropy also had an 8% increase in score when the considered cameras increased from 50 to 100, but requires more than doubling

Metric	5 to 10 Cameras	10 to 20 Cameras	20 to 50 Cameras	50 to 100 Cameras
Data Entropy	0.17%	0.03%	0.09%	0.03%
DDS Entropy	0.08%	12%	2.3%	1.6%
Depth Entropy	5.6%	1.3%	11.1%	8.7%
Max Depth	2.2%	1.4%	11.6%	0.0%
PB	1.5%	0.0%	0.0%	0.0%
Projected Area	1.3%	0.0%	0.0%	0.0%
Shading Entropy	0.07%	4.7%	2.6%	3.6%
Viewpoint Entropy	1.1%	1.2%	0.7%	1.3%
Visibility Ratio	1.2%	0.3%	1.1%	0.8%
Visible Triangles	2.0%	0.7%	3.0%	0.8%
VKL	0.2%	0.4%	0.1%	0.1%

Table 14. The average percentage increase of each metric's viewpoint quality score as budget increases using Diagonal 43. These averages are calculated over all 100 cycles.

the in situ execution time. Shading Entropy also benefited from increasing the camera budget, but to a lesser extent: experiencing a 4.7% increase when budgeted cameras went from 10 to 20, and a 3.6% increase when budgeted cameras went from 50 to 100, but, again, results in an increase of in situ execution time. DDS Entropy also experienced similar benefits: in particular a 12% increase when the camera budget was increased from 10 to 20 cameras, as well as percentage increases of 2.3% and 1.6% when the budget was increased from 20 to 50 cameras and 50 to 100 cameras, respectively. But for the seven other VQ metrics, they experienced very little benefit when increasing the number of budgeted cameras, and certainly not enough to offset the necessary the increase in execution time. One reason for this could be that for most of these metrics, a quality viewpoint is easy to find, so increasing the budget may not necessarily yield a significantly better viewpoint. Or alternatively, a quality viewpoint may be hard to find, as some heatmaps from Figure 24 and Table 11 would suggest for individual VQ metrics. The percentage increases, or lack thereof, reinforces the conclusions from Section 4.3.2: that, depending on the VQ metric, the search algorithms can find a viewpoint with a

reasonably high VQ metric score quite quickly, but to improve upon a metric score may take longer than in situ constraints allow.

4.4 Conclusion

The goal of this research was to show that in situ automatic camera placement is viable and that good viewpoints can be found quickly. This work is the first to parallelize these VQ metrics and optimize them for both shared- and distributed-memory environments. To show the viability of in situ implementation we ran performance study that shows VQ metrics are much less costly compared to the rendering process. This work also introduced several search algorithms and studied their behavior, first in a post-hoc environment, using processing resources that would be unavailable in an in situ environment. From this preliminary phase, we found that, depending on the VQ metric, good viewpoints for cycles of scientific data sets are not sparse nor hard to find. The search algorithms can find a viewpoint with a metric score in the $80^{th} - 85^{th}$ percentile rather quickly, but finding a viewpoint in the 90^{th} percentile would require a rendering budget that is likely infeasible for in situ execution. Further, previous work [92] has shown that while users dislike low scoring viewpoints, they often disagree on the highest scoring viewpoints. As a result, we believe the best in situ strategy is to quickly find a viewpoint that is "good" rather than taking longer to find a viewpoint that is the "best." Our belief in this strategy aligns with our findings, in that increasing camera budget led to only modest changes in VQ metric score. Overall, few metrics benefited significantly when the camera budget was increased to 50 or 100 cameras, with most metrics experiencing the greatest score increase when considering 20 cameras.

Future work is two-fold: (1) examining the behavior of optimal camera placement over time and (2) design and evaluate best practices for how often a search should be conducted. In regards to the former point, we plan to analyze optimal viewpoint placement over time in order to determine the in situ behavior of optimal viewpoints for a given scientific simulation. Based on these results, we will be able to design a fixed search interval, that searches for a new optimal viewpoint after a fixed number of time steps, as well as design a trigger, that when activated will immediately begin a new search for an optimal viewpoint search. These research directions are explored in the next chapter.

CHAPTER V

OPTIMAL VIEWPOINT PLACEMENT OVER TIME (OVPOT) FOR SCIENTIFIC SIMULATIONS

This chapter is unpublished co-authored work. Contributions to this work are the following: I am the first author of this work. I coded the majority of this project's code base, designed the project's experiment parameters, executed the study's runs on the Summit supercomputer and analyzed the results. I also wrote the entirety of the manuscript. Meghanto Majumder executed the study's runs on CDUX's (Computing and Data Understanding at eXtreme Scale) local computer cluster, Alaska, and analyzed the results. Hank Childs provided extensive feedback over the course of this work and helped in editing the manuscript.

5.1 Introduction

Previous research has explored using Viewpoint Quality (VQ) metrics as a way to measure the quality of a camera placement. For the most part, the application of these metrics have been limited to individual time steps, rarely taking into account time-varying data. When considering the study of optimal camera placement over time, there are several research gaps. While previous work has explored the optimal camera budget when searching for a new camera for individual cycles, there has been no investigation into the stability of the goodness of a camera placement as a simulation evolves. Additionally, there has been no investigation into the frequency of searching for a new camera placement. This chapter aims to fill these research gaps with two research contributions:

– We determine the best camera budget to use for time-varying data.

– We consider how our findings can be implemented as a "trigger' based method that will only conduct a new camera search when the current best viewpoint falls below some user-defined threshold.

5.2 Related Work

The key themes of this work, and the topics of this section, involves triggers 5.2.1 and automatic camera placement for time-varying data 5.2.2. Automatic camera placement for individual cycles is also a key theme for this work, and was already described in Chapter II.

5.2.1 Triggers. Triggers are a powerful mechanism that can guide the execution of algorithms, determining when to perform actions and what actions to perform. Past research for triggers fall into one of two categories: domain agnostic and domain specific.

5.2.1.1 Domain Agnostic Triggers. Domain agnostic triggers are typically applied to arrays of values, and do not need special knowledge of the simulation or domain. there have been several notable works in this space. Ling et al. [83] uses feature importance metrics and machine learning to determine when data should be saved to disk. Yamaoka et al. [146] calculates the Kullback-Leibler divergence between PDFs of the simulation's timesteps, and uses the results to adapt the sampling rate for in situ data reduction. Larsen et al. [74] provides user-defined triggers via a visualization service.

5.2.1.2 Domain Specific Triggers. Domain specific triggers are similar to domain agnostic triggers, in that they are applied to arrays of values, but they differ in that they require special knowledge of the simulation or domain. Bennett et al. [18] developed a trigger that detects the ignition in combustion simulations; this work was made more robust by Salloum et al. [113]. Work from

Ullrich and Zarzycki [133] and Zhao et al. [150] created triggers for tropical storms within climate simulations. Additionally, work from Liu et al. [85] and Sun et al. [125] developed triggers for eddy identification and tracking. These are just a handful of works that involve anomaly detection, an area of research that has developed a number of both domain agnostic and specific solutions [33, 112, 4]. In a lot of cases, anomaly detection provides domain agnostic triggers, but are made domain specific when tailored to a particular application.

5.2.2 Camera Placement Over Time. Several flow Visualization techniques have attempted to optimize data visualization as a simulation progresses. Lee et al. [76] was the first to consider seed placement and viewpoint selection simultaneously, utilizing the entropy of the vector field to determine optimal camera placement and then determining the best seed placements for a given viewpoint, but does not alter the viewpoint over time. Tao et al. [129] uses an information channel between streamlines and viewpoints to choose streamlines that are view independent and then generates a path between the selected viewpoints. Ma et al. [88] chooses streamlines that are view dependent and subsequently developed FlowTour which automatically determines critically regions present in steady flow fields and then temporally traverses the viewpoints that best display these regions. This work was then extended to not only consider steady flow fields but unsteady flow fields as well [87].

Camera placement over time has also been applied to in situ visualization via Computational Steering [52, 110, 53, 46, 123, 15, 98, 50, 49, 57, 60, 144, 42, 40, 8], a tool that allows users to interact with a simulation in situ, such as changing viewpoint, without having to write to disk. This differs from our own work on

automated placement, since computational steering enables domain scientists to set camera positions via human-in-the-loop interactions.

Several works have tried to determine the best view path for time-varying datasets. Bai et al. [11] develop a view path for volume data by measuring feature evolution and determining a viewpoint's quality based on the topological and visual features. Ji and Shen [66] provided a more static approach by determining each cycle's optimal viewpoint and then employ dynamic programming to design a view path.

Another research strategy involves writing informative videos to disk. Yamamoto et al. [145] output videos of the simulation as it evolves over time from a number of different viewpoints. As the simulation progresses, data is passed to a rendering process via a "Membrane Layer," which separates the running simulation from the rendering process. But if the simulation generates data faster than it can be rendered it will overwrite the data that has not yet been written to disk, users can then access a movie database to view the rendered data over time.

5.3 Our Method

This section describes our method for constructing and evaluating an algorithm for determining the optimal viewpoint placement over time (OVPOT).

The central question of our OVPOT algorithm is whether or not to search for a new camera position at a given cycle. The OVPOT algorithm will use Viewpoint Quality (VQ) metrics to evaluate viewpoints and decide the best view from a sampling of camera placements. In particular, this work utilizes DDS Entropy, the VQ metric from Chapter III to produce views most preferred by visualization experts and domain scientists. To determine camera placement, our algorithm utilizes a Fibonacci's Lattice in order to equally space camera placements

around the data set; this is similar to spacing camera placements along the diagonal of the data (XZ-plane), which was shown in Chapter IV to find quality views fastest.

The results from Chapter IV showed that searching for a quality camera placement can be costly. Further, the VQ metric DDS Entropy is an imperfect oracle when choosing between good views, meaning there is no guarantee that the view with the highest score will actually be the most preferred. Thus, we favor limiting the total number of searches that are performed in situ so as not to unnecessarily encumber the simulation. More so, scientific data is typically considered to be smooth and have temporal-coherence, meaning data changes gradually as the simulation progresses, so performing frequent camera searches would most likely lead to superfluous and repetitive results. Given the nature of the data, our trigger criteria is based on the notion that a quality camera will remain a quality camera until determined otherwise. Our OVPOT algorithm only has one criteria that will trigger a search for a new camera placement: the VQ metric score of the current best camera position has dropped below a percentage threshold.

In order to develop an algorithm that can output the the best camera position over time, we need to optimize (1) the camera budget used when searching for a new camera and (2) the percentage threshold that is used to trigger a new camera search. The construction and analysis of our OVPOT algorithm was conducted in two phases: Phase 1 involves post-hoc analysis of view placements over time and Phase 2 incorporates the results of Phase 1 into in situ execution and evaluates the performance of different percentage thresholds. This section is organized accordingly.

5.3.1 Phase 1: Post-hoc Analysis. Before implementing and evaluating an OVPOT algorithm in situ, we first analyze the behavior of camera positions over time for scientific simulations. Evaluating a VQ metric for a given camera position has non-negligible cost, meaning it is very desirable for an OVPOT algorithm to produce good views with a "small" number of VQ metric evaluations. The goal of Phase 1 is to utilize post-hoc processing to inform best practices for Phase 2's in situ implementation, namely, the optimal camera budget for time-varying data.

For a number of small- and medium-sized simulations, we will evaluate the effects of using variable camera budgets for time-varying data has on the quality of the generated viewpoint. To do this, for each simulation run, at some fixed-interval, we evaluate the VQ metric score for 100 camera placements . Then, we evaluate the average maximum score attainable when considering X number of cameras. That is, for X number of cameras, we considered every possible set of X cameras that are equally spaced among the total 100 cameras, then, amongst these sets, we average the maximum VQ metric score. For example, if we were considering a camera budget of four, from the total 100 cameras we would create sets containing four camera in the following way: $[0, 25, 50, 75], [1, 26, 51, 75], \ldots, [24, 49, 74, 99]$. We then compute the average maximum VQ metric score from all of these sets. We computed this value for the duration of the simulation.

From these experimental runs we can analyze the effect a camera budget has on maximum score over time and determine an appropriate in situ camera budget for time-varying data, and subsequent use in Phase 2.

5.3.2 Phase 2: In Situ Implementation and Evaluation. The goal of Phase 2 is to use the results from Phase 1 for in situ implementation

and evaluate different triggers for time-varying data. We perform several in situ evaluations of medium- and large-scale scientific simulations. The runs consider the recommended camera budgets from Phase 1 and evaluate different trigger thresholds. The trigger thresholds are used to determine if a search for a new "best" camera placement needs to be conducted and are based on a percentage of the VQ metric score of the current best camera.

Our OVPOT algorithm behaves as follows: At a fixed-interval, each time our algorithm is executed, it will evaluate the the current best viewpoint and determine if a new camera search is triggered. For example, assume we execute our OVPOT algorithm every cycle and the trigger threshold is set as 95% of the best camera. Further, assume cycle T_1 requires a new camera search, and this search determines that camera C_1, with a VQ metric score M_1, has the highest score amongst all considered cameras.Then, for the next four cycles the metric score for camera C_1 continually decreases until timestep T_5 when the metric score, M_5, for camera C_1 falls below the threshold, that is, $M_5 < .95 * M_1$. At this point, a search for a new camera is triggered and the camera with the highest metric score is chosen as the new "best" viewpoint.

From these experimental runs we can analyze the effect trigger thresholds have on camera search frequency and how best to optimize automatic in situ camera placement over time.

5.4 Experimental Overview

Studying optimal viewpoint placement over time was conducted in two phases. Phase 1 is a post-hoc evaluation of variable camera budgets over time. Phase 2 is an in situ study on the frequency of a trigger-based search routine. The

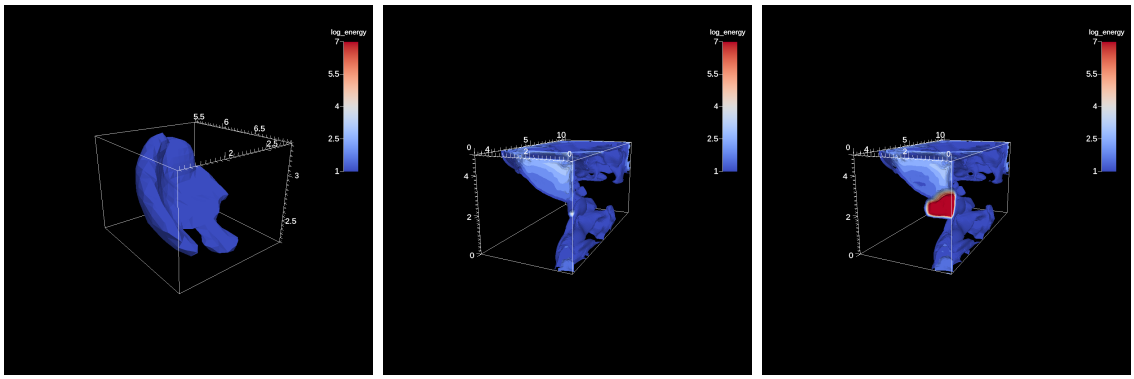


Figure 26. Renderings of the Impact input deck over time. The field, log of energy, was transformed using isosurfaces as well as a clipping of one of the quadrants.

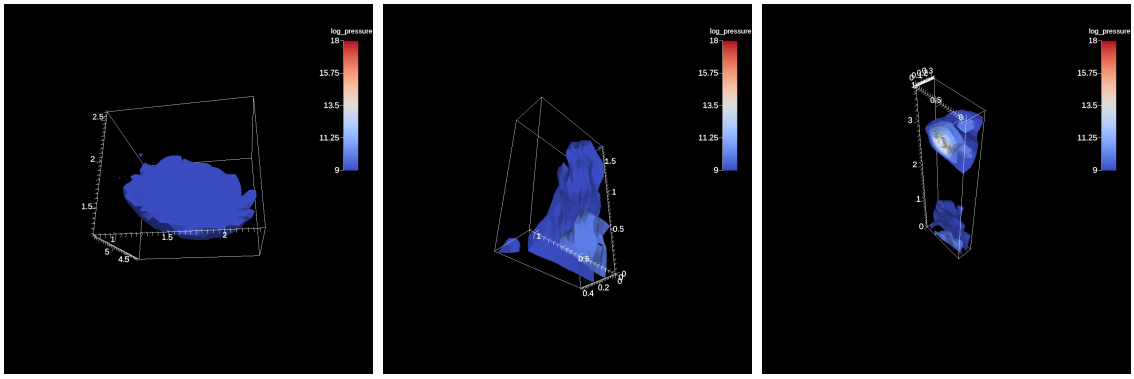


Figure 27. Renderings of the Impact input deck over time. The field, log of pressure, was transformed using isosurfaces as well as a clipping of one of the quadrants.

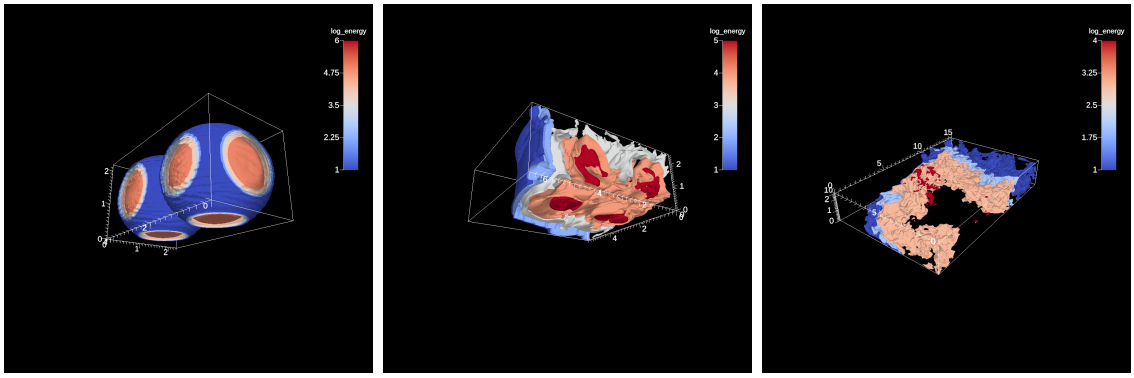


Figure 28. Renderings of the Ball of Fury input deck over time. The field, log of energy, was transformed using isosurfaces.

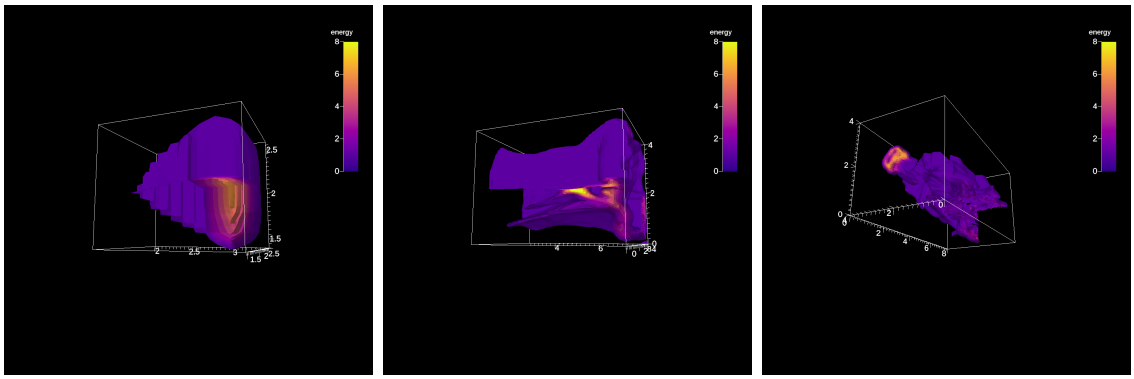


Figure 29. Renderings of the Jetbox input deck over time. The field, energy, was transformed using isosurfaces as well as a clipping of one of the quadrants.

rest of this section is organized accordingly and details the experimental overview of these phases.

5.4.1 Phase 1: Post-hoc Experiment. Phase 1 was conducted on Alaska, a computer-cluster located at the University of Oregon that utilizes Intel XEON CPUs. The executed simulation was CloverLeaf3D, a simulation that is provided in Ascent [71] that utilizes input decks to define the problem setup. For this study we utilized two different input decks and evaluated the optimal camera placement over time for a chosen field. We utilized the VQ metric DDS Entropy to determine the quality of the camera placement. The camera placements were chosen using the Fibonacci Lattice, a method for equally distributing points around a sphere. Whenever a new camera search is performed, 100 camera placements are evaluated using the chosen VQ metric. Additionally, the simulation data is transformed into an isosurface to better display the data.

The three configurations, meaning input decks, field, and execution parameters, are as follows:

– Configuration #1 used the "Impact" input deck and visualized the energy field. This simulation was executed for 830 cycles and a new camera search is

performed every 10^{th} cycle. The size of this simulation was 64^3 and executed on a single node. Renderings of this simulation are shown in Figure 26.

- Configuration #2 used the "Impact" input deck and visualized the pressure field. This simulation was executed for 830 cycles and a new camera search is performed every 10^{th} cycle. The size of this simulation was 64^3 and executed on a single node. Renderings of this simulation are shown in Figure 27.

- Configuration #3 used the "Ball of Fury" input deck and visualized the energy field. This simulation was executed for 9,100 cycles and a new camera search was performed every 100^{th} cycle. The size of this simulation was 64^3 and executed on a single node. Renderings of this simulation is shown in Figure 28.

5.4.2 Phase 2: In Situ Experiment. Phase 2 was conducted on both Alaska and the OLCF supercomputer Summit. Once again, the simulation data is transformed into an isosurface before being evaluated by our OVPOT algorithm.

On Alaska, CloverLeaf3D was executed with the following two configurations:

- Configuration #1 used the "Jetbox" input deck and visualized the energy field. This simulation was executed for 1,200 cycles and a new camera search was performed every 10^{th} cycle. The size of this simulation was $256 \times 128 \times 256$ and executed on a single node. In addition to being transformed into an isosurface, this data also clips one quarter of the data, as shown in Figure 29. For each simulation run we utilized one of two camera budgets: 10 and 20 camera samples with viewpoints chosen using the Fibonacci Lattice. And for

each camera budget we evaluated three trigger thresholds that are based on a percent of the best camera's metric score. The threshold percentages used are: 95%, 96%, 97%, 98%, 99%, and 100%.

- Configuration #2 used the "Ball of Fury" input deck and visualized the energy field. This simulation was executed for 9,100 cycles and a new camera search was performed every 100^{th} cycle. The size of this simulation was 64^3 and executed on a single node, renderings of this simulation are shown in Figure 28. For each simulation run we utilized one of two camera budgets: 10 and 20 camera samples with viewpoints chosen using the Fibonacci Lattice. And for each camera budget we evaluated three trigger thresholds that are based on a percent of the best camera's metric score. The threshold percentages used are: 95%, 97%, and 99%.

On the Summit supercomputer, the AMR-Wind simulation was excuted with the following parameters:

- The size of this simulation was 848^3 and ran on 25 nodes using 130 MPI ranks and 130 GPUs. The simulation was executed for 100 cycles and a new camera search was performed every cycle.For each simulation run we utilized one of two camera budgets: 10 and 20 camera samples chosen using the Fibonacci Lattice. And for each camera budget we evaluated three trigger thresholds that are based on a percent of the best camera's metric score. The threshold percentages used are: 95%, 96%, 97%, 98%, 99%, and 100%.

The chosen VQ metric for all of these simulation runs is DDS Entropy.

5.5 Results

This section details the results of Phase 1 and Phase 2 described in Section 5.3 with experiment parameters from Section 5.4.

5.5.1 Phase 1: Post-hoc Results. To examine the impact camera sample size has on VQ metric score over time, we evaluated the following camera budgets: 1, 4, 5, 10, 20, 25. That is, for all of the camera budgets, from the total 100 cameras sampled, we created sets of equally spaced cameras for each of the specified camera budgets. Figure 30 shows the results of the tradeoff between the number of cameras considered and the maximum attainable VQ metric score. The top row of figures shows the log of the average maximum score over time for a given camera budget in comparison to the maximum score achieved when considering all 100 cameras. The bottom row of figures shows the relative performance of each budget of cameras when scaled against the maximum score out of all 100 cameras. These figures show that, over time, considering a single camera greatly limits a user's ability to find a high scoring camera placement. Notably, there is a significant improvement when increasing the camera budget from 1 to 4 or 5 cameras. Additionally, there is an increase in metric score when the budgets are increased to 10, 20, and 25 cameras, albeit less so. But overall, the average maximum score for budgets 10, 20, and 25 are consistently high and, going forward, we recommend considering a minimum number of 10 cameras (or possibly 5 if constraints are tight) and a maximum of 20 cameras. We do not recommend a camera budget of 25 because our results show that using a camera budget of 20 over time achieves comparable while considering 20% fewer cameras.

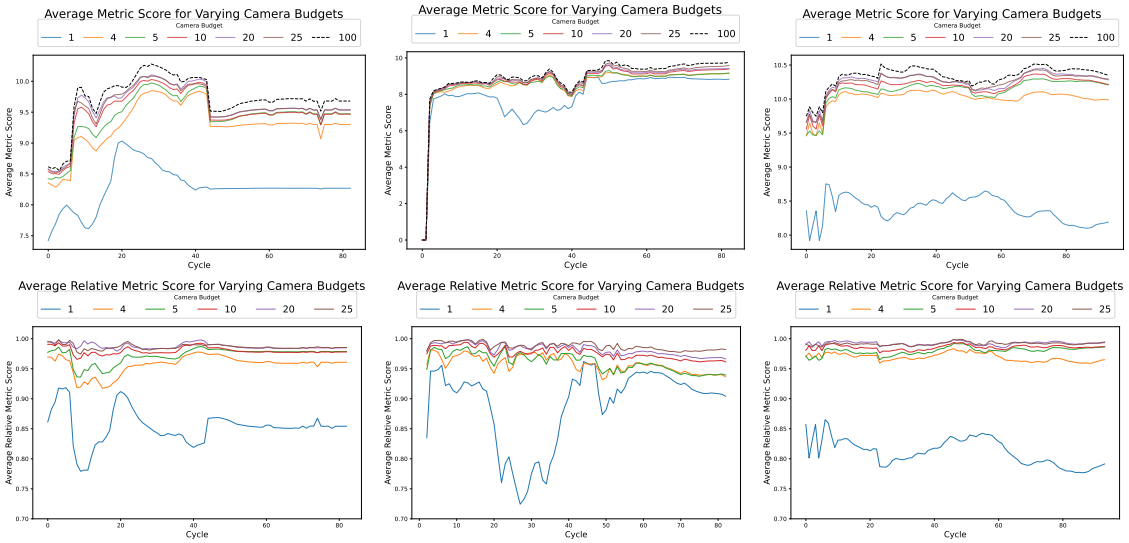


Figure 30. The average maximum score for each camera budget over time. The top row shows the log of average metric score for each of the considered camera budgets, with the dotted black line showing the maximum metric score that was achieved when considering all 100 cameras. The bottom row shows the relative maximum average score for each camera budget. From left to right the data sets and respective fields are as follows: Impact (field: log of energy), Impact (field: log of pressure), Ball of Fury (field: log of energy). The metric used is DDS Entropy from Chapter III.

5.5.2 Phase 2: In Situ Results.

To determine how often to search for a new camera placement, this phase executes several simulations with variable percentage thresholds and analyzes its effects on the frequency of searches.

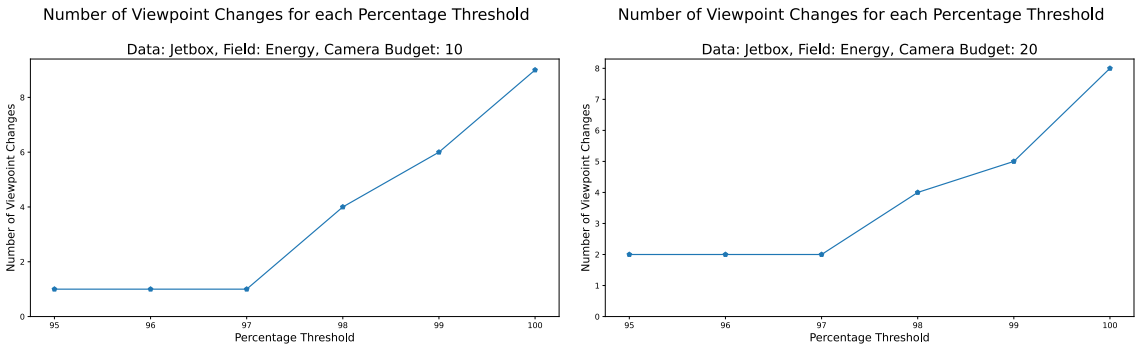
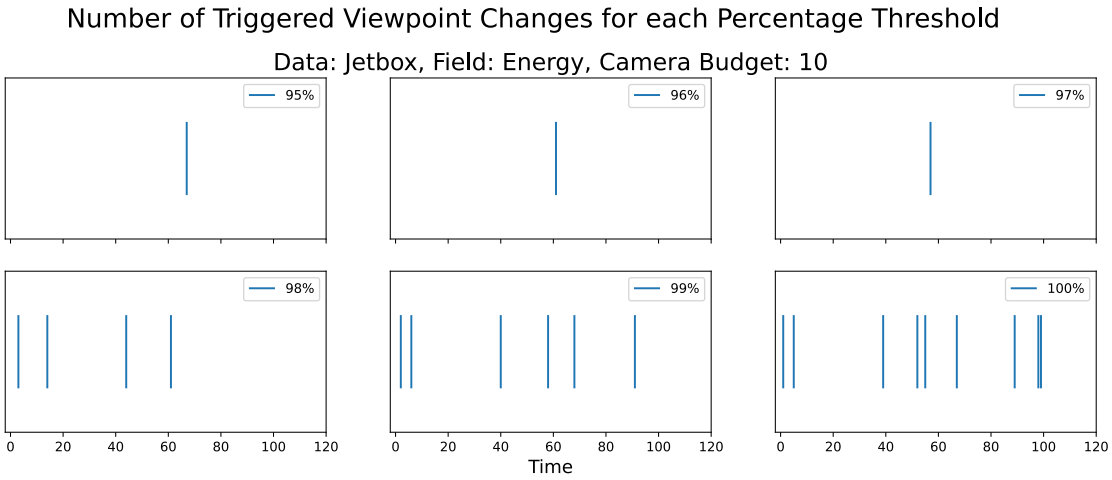


Figure 31. The number of in situ viewpoint changes that were triggered for each percentage threshold over the course of the Jetbox simulation.



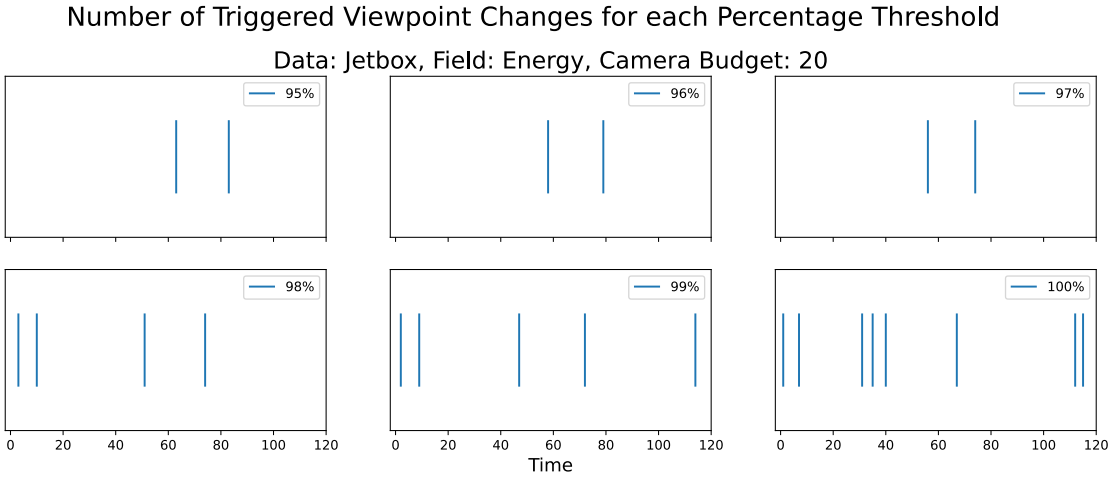


Figure 32. The number of in situ viewpoint changes for each percentage threshold and when they were triggered over the course of the Jetbox simulation. Note that a vertical blue line means that a new search was triggered, and the Y-axes are unused in these plots.

Figures 31 - 34 show the in situ results using both camera budgets for the Jetbox CloverLeaf3D input deck. Figure 31 shows the number of times each percentage threshold changed viewpoints over the course of the simulation. When using a camera budget of 10, the percentage thresholds set to 95%, 96%, and 97% were only triggered once over the course of the simulation. Larger thresholds,

unsurprisingly, experienced a greater number of viewpoint changes: the 98% and 99% threshold were triggered four and six times, respectively, which are both a reduction from the nine triggers experienced with a threshold of 100%. Results for Jetbox with a camera budget of 20 are similar: the 95%, 96%, and 97% thresholds were triggered two times, the 98% threshold was triggered four times, the 99% threshold was triggered five times, and the 100% threshold was triggered eight times over the course of the simulation. And Figure 32 shows when, over the course of the simulation, the percentage threshold was triggered and activated a new camera search. For all trigger thresholds, the frequency of viewpoint changes is consistent across camera budgets. Figure 33 graphs the difference between the metric score of the chosen camera and the metric score of the optimal camera over time for all thresholds.. We see that, no matter the budget, the chosen camera score can deviate from the optimal camera score, but it will trigger a new camera search if it deviates too far from optimal. Overall, the metric scores produced by each threshold, while not optimal, are not far off, producing, on average, relative metric scores that are in the top 99^{th} percentile, as in shown in Figure 34.

The results from Ball of Fury, plotted in Figures 35 - Figures 38, shows similar results to Jetbox, but with even fewer viewpoint changes. With a budget of 10 cameras, the 95% and 97% thresholds both experienced zero viewpoint changes, meaning the camera chosen on the first cycle was the chosen camera for the entire simulation run, and the 99% threshold was triggered only three times. The frequency of viewpoint changes for Ball of Fury increased when the camera budget was 20: the 95% threshold was triggered once, the 97% threshold was triggered four times, and the 99% threshold was triggered five times, as shown in Figure 35. The difference in metric scores for the chosen camera and optimal camera for all

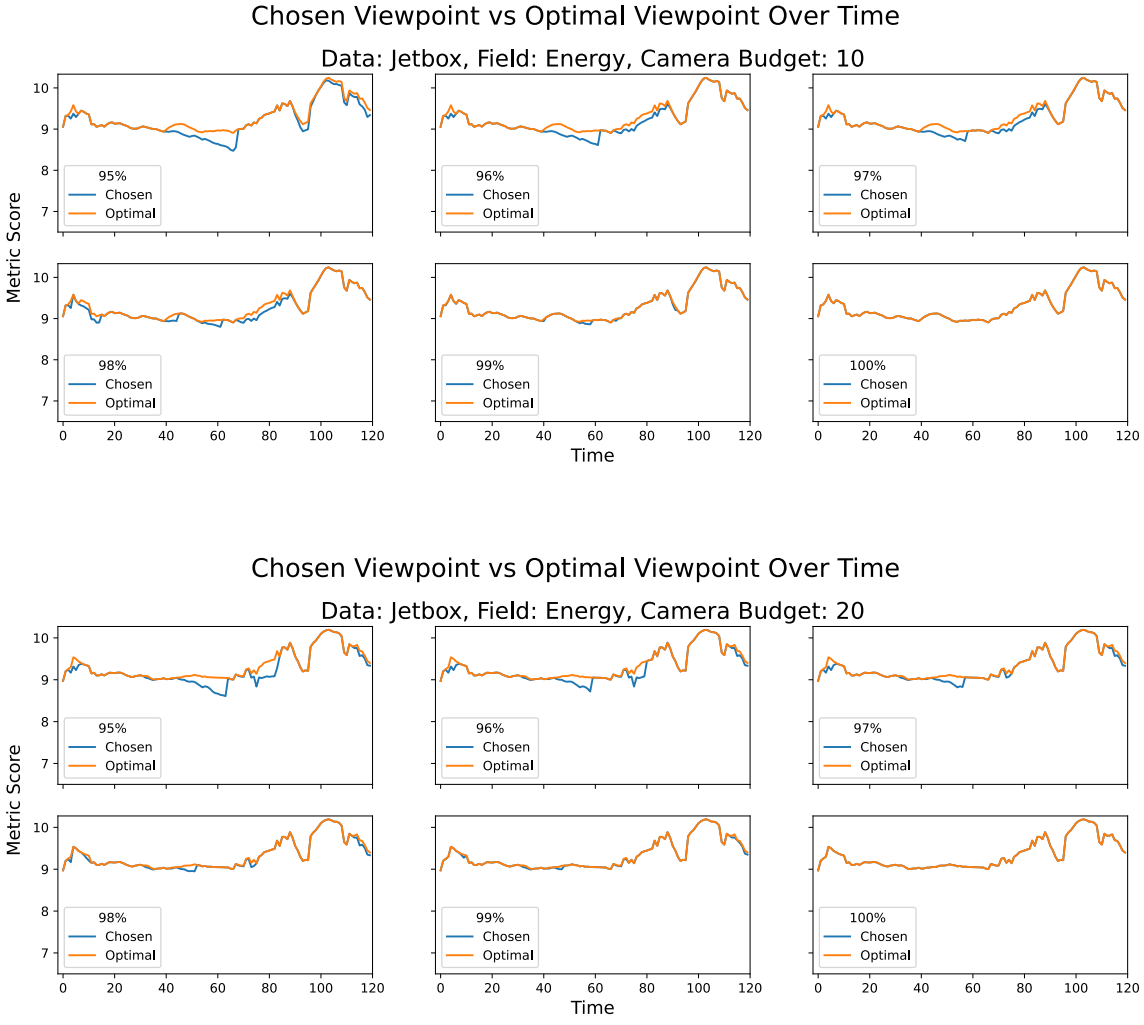


Figure 33. For each camera budget and threshold trigger, we graphed the scores of the camera placement chosen in situ against the top scoring camera placement over time for the Jetbox simulation. The orange line in each graph represents the top scoring camera placement amongst each camera budget. The blue line in each graph represents the score of the chosen camera placement when executing our OVPOT algorithm. The graphs show that the chosen camera placement may be sub-optimal, and a larger camera budget may cause greater deviation for optimal, but this deviation will never exceed the $X\%$ where $X = 100 -$ chosen percentage threshold.

thresholds is shown Figure 37. Much like Jetbox, we see only slight deviations from the optimal camera placement before a new search is triggered, but we do

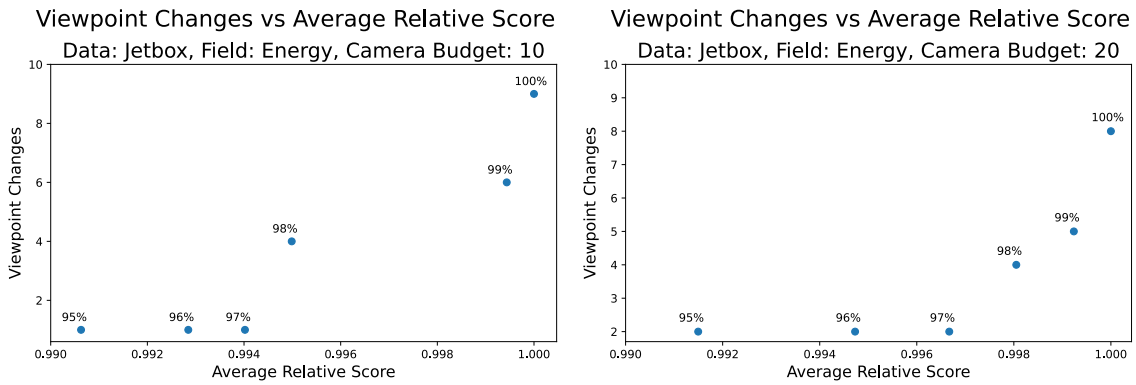


Figure 34. The average relative metric score of the chosen camera placements for the chosen camera budgets and trigger thresholds for the Jetbox simulation.

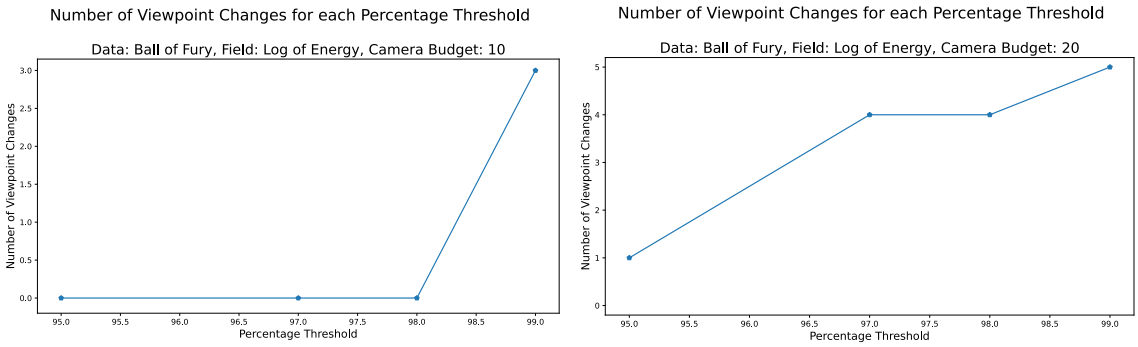


Figure 35. The number of in situ viewpoint changes that were triggered for each percentage threshold over the course of the Ball of Fury simulation.

see more deviation when using a camera budget of 20 versus 10. For this data set, increasing the camera budget decreased the relative quality of the chosen camera to a higher degree. For a camera budget of 10, all thresholds achieve an average relative metric score in the top 99^{th} percentile. Whereas, for a camera budget of 20, only the 97%, 98% and 99% thresholds yield cameras with an average relative score in the 99^{th} percentile, the 95% threshold yields a camera that over time only achieves an average relative score in the 98^{th} percentile, as shown in Figure 38.

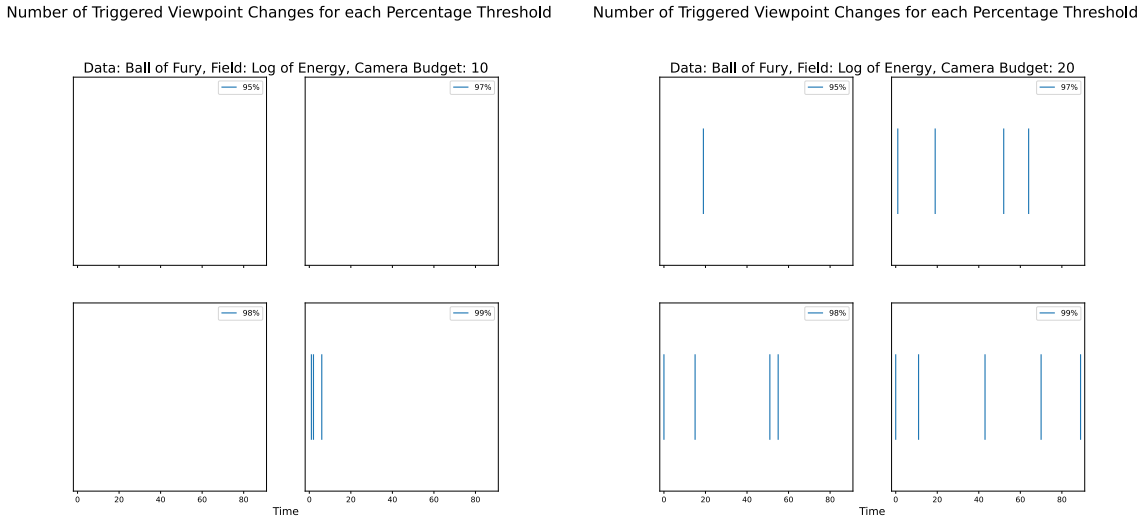


Figure 36. The number of in situ viewpoint changes for each percentage threshold and when they were triggered over the course of the Ball of Fury simulation. Note that a vertical blue line means that a new search was triggered, and the Y-axes are unused in these plots.

While, AMR-Wind triggered more searches over the course of the simulation than any other data set, as shown in Figure 39, it also experiences a similar reduction in search frequency via threshold triggers. Overall, the results show consistent search frequency across camera budgets and thresholds, as shown in Figure 40, which plots the frequency of triggering a new camera search, and shows that all thresholds for both camera budgets experience a clustering of searches in the early cycles. Much like Ball of Fury, the larger camera budget provides a wider array of viewpoints and, thus, a potential for greater deviation from the optimal camera, as shown in Figure 41. This deviation can also be seen in Figure 42 which plots each threshold's average relative score. All thresholds when using a camera budget of 10 produce scores that average in the 99^{th} percentile, whereas only the 95% and 99% thresholds produce scores that average in the 99^{th} percentile when using a camera budget of 20.

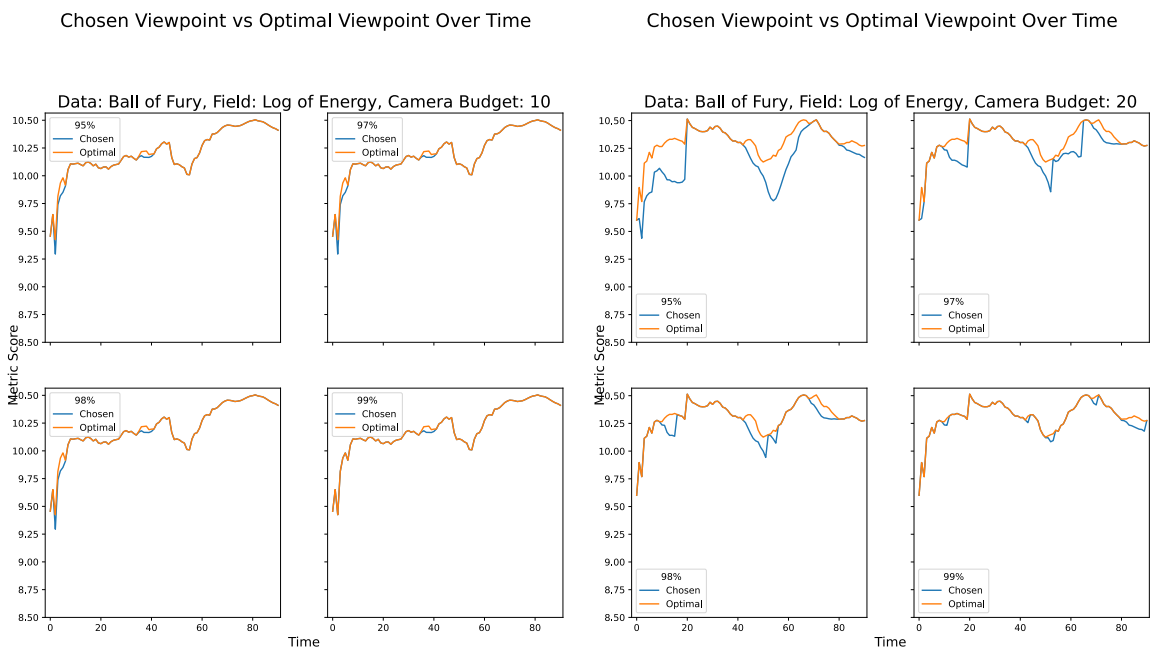


Figure 37. For each camera budget and threshold trigger, we graphed the scores of the camera placement chosen in situ against the top scoring camera placement over time for the Ball of Fury simulation. The orange line in each graph represents the top scoring camera placement amongst each camera budget. The blue line in each graph represents the score of the chosen camera placement when executing our OVPOT algorithm. The graphs show that the chosen camera placement may be sub-optimal, and a larger camera budget may cause greater deviation for optimal. That said, this deviation will never exceed the $X\%$ where $X = 100 -$ chosen percentage threshold.

When it comes to frequency of searches, all data sets experienced a reduction in viewpoint searches when utilizing our triggered-based OVPOT algorithm. In most instances, reducing the percentage threshold significantly reduced the number of searches performed over time. For nearly all the data sets, employing a trigger threshold of 97% or lower reduced the number of searches by 50% or more.

5.6 Conclusion

This chapter explores the best practices for optimizing camera placement over time. Searching for a quality viewpoint can be costly, it is important that our

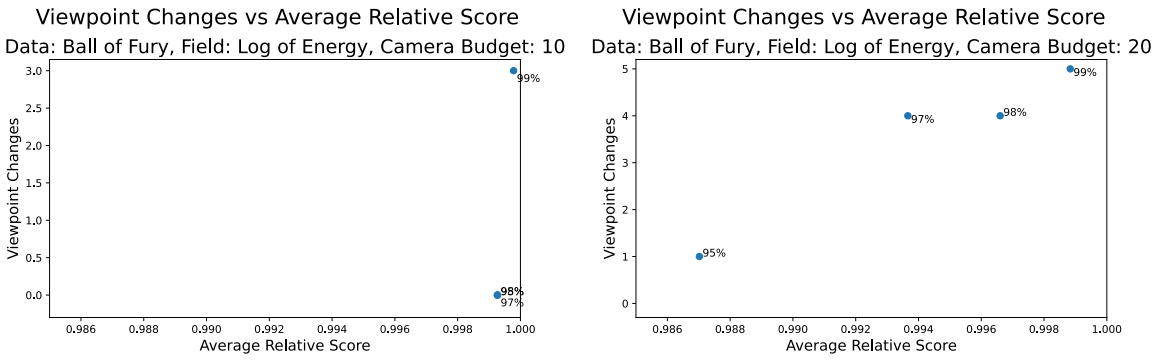


Figure 38. The average relative metric score of the chosen camera placements for the chosen camera budgets and trigger thresholds for the Ball of Fury simulation.

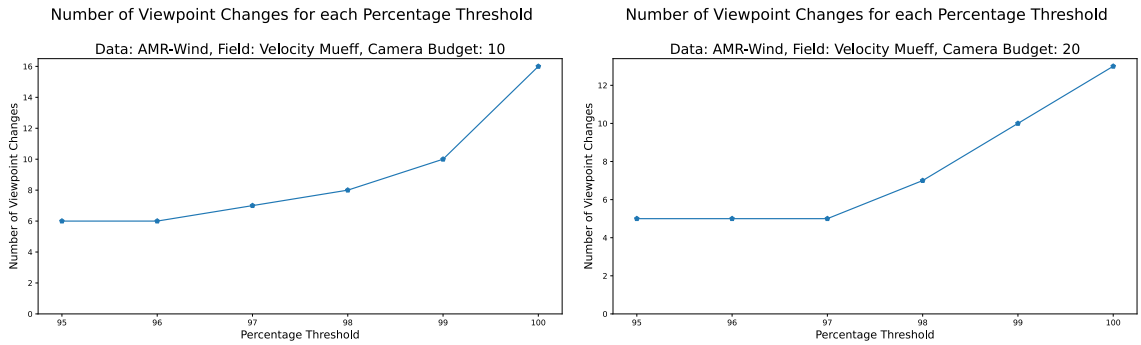


Figure 39. The number of in situ viewpoint changes that were triggered for each percentage threshold over the course of the AMR-Wind simulation.

OVPOT algorithm minimizes the number of searches conducted over the course of the simulation. With this work, we evaluate variable camera budgets over time, we show that when searching for a new camera, a user should employ a camera budget of 10 to 20 cameras. We also explored the use of a trigger threshold that is based on a percentage the best camera's VQ metric score. A new camera search will be triggered when the current best camera falls below a selected percentage threshold. The use of this trigger minimizes the number of searchers over the course

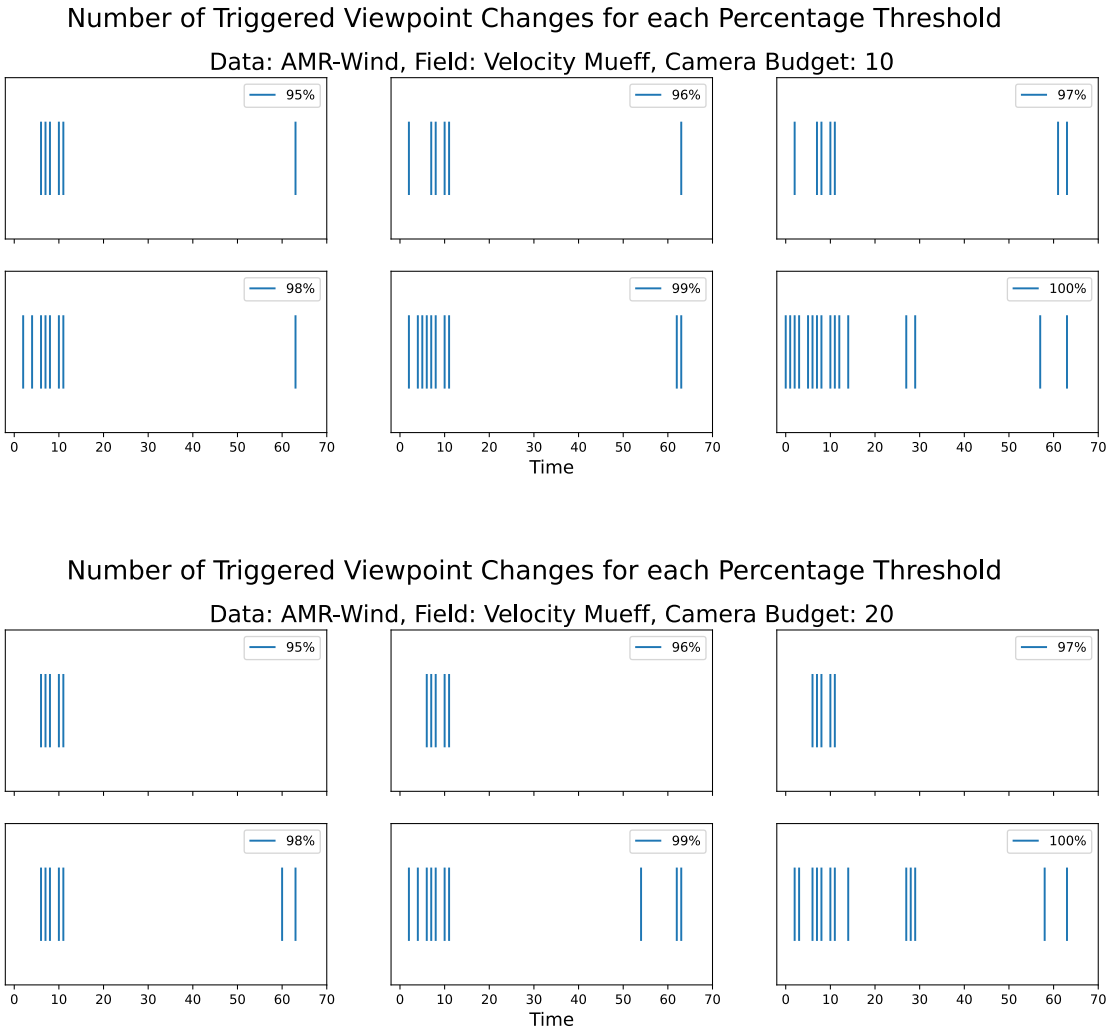


Figure 40. The number of in situ viewpoint changes for each percentage threshold and when they were triggered over the course of the AMR-Wind simulation. Note that a vertical blue line means that a new search was triggered, and the Y-axes are unused in these plots.

of the simulation by limiting it to time cycles that represent change. This research found that using a percentage threshold can significantly reduce the frequency of in situ camera searches, finding that using a threshold of 97% or lower can reduce the number of searches by more than 50% in most cases. Overall, this work informs on the best practices for optimizing automatic in situ camera placement for time-

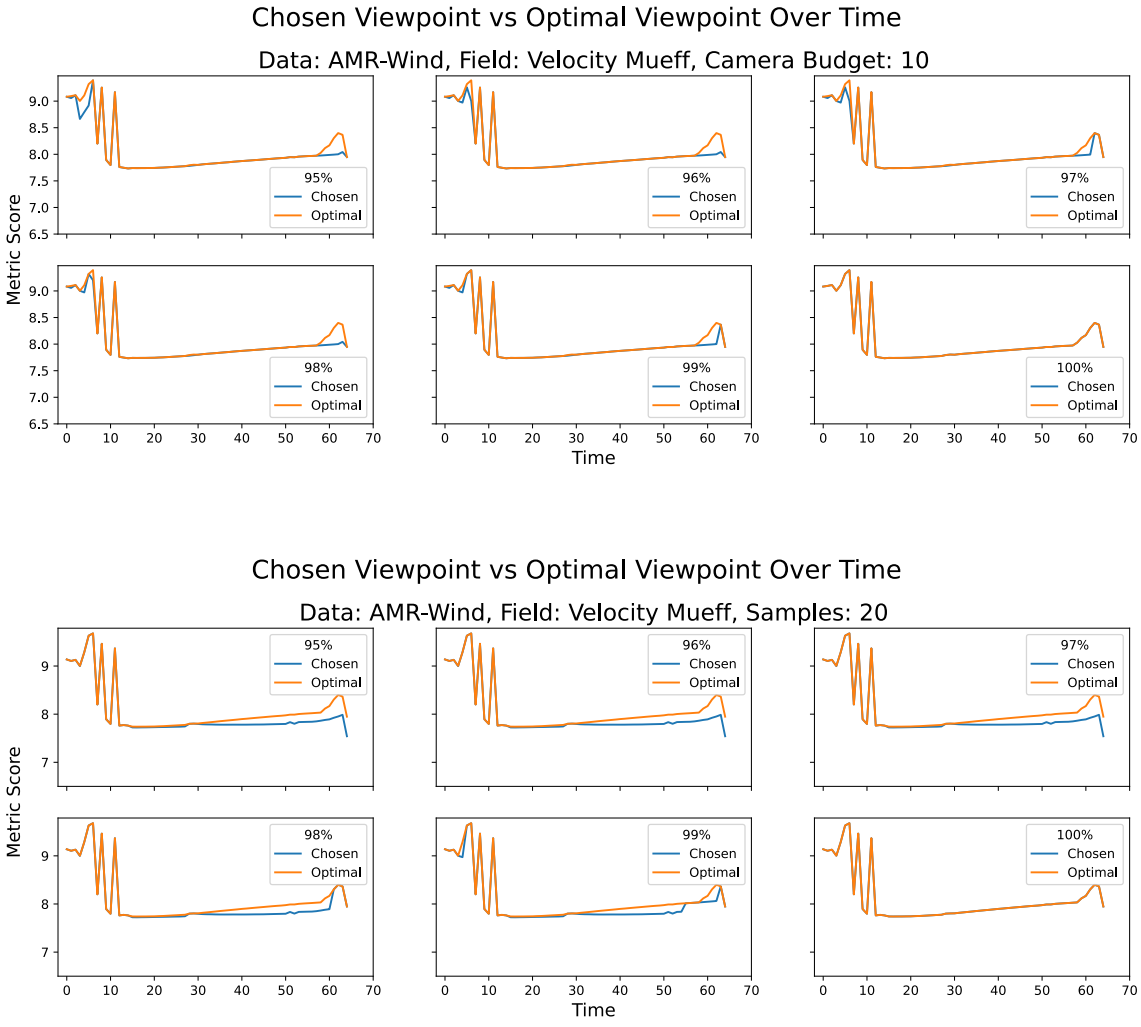


Figure 41. For each camera budget and threshold trigger, we graphed the scores of the camera placement chosen in situ against the top scoring camera placement over time for the AMR-Wind simulation. The orange line in each graph represents the top scoring camera placement amongst each camera budget. The blue line in each graph represents the score of the chosen camera placement when executing our OVPOT algorithm. The graphs show that the chosen camera placement may be sub-optimal, and a larger camera budget may cause greater deviation for optimal, but this deviation will never exceed $X\%$ where $X = 100 -$ chosen percentage threshold.

varying data. This work balances the cost of searching with for a quality viewpoint against the overall quality of a viewpoint over the course of a simulation. It also

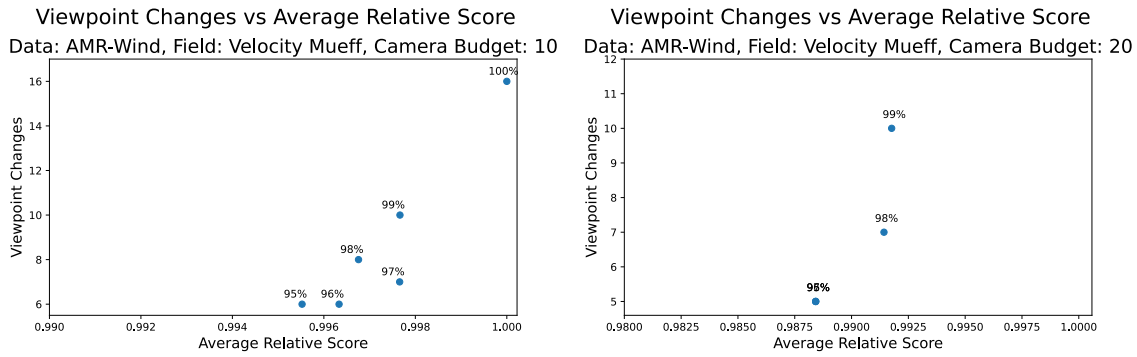


Figure 42. The average relative metric score of the chosen camera placements for the chosen camera budgets and trigger thresholds for the AMR-Wind simulation.

presents a percentage-based trigger to limit the frequency of searching for a new viewpoint.

Future work for this research includes evaluating our OVPOT algorithm on more simulations, as well as a more thorough investigation into the effects that OVPOT execution frequency has on camera search frequency. Additionally, in the circumstance where entropy continually increases as the simulation progresses, it may be beneficial to perform a new camera search at fixed intervals regardless of the trigger. Further, rather than outputting just an image, a video that traverses an informative path from the old camera to the new camera would better show the change that just occurred within the simulation.

CHAPTER VI

CONCLUSION

I was the primary author of this manuscript, and Hank Childs provided editorial suggestions.

With the rise of in situ processing, analysis and visualization is routinely being executed without a human in the loop (HITL). Camera placement is one of those tasks traditionally performed in a post-hoc, HITL fashion. The goal of this dissertation was to research methods for researchers running large-scale scientific simulations to automatically determining camera placement in situ. This work has shown that to be the case with each research thrust that answered our four research subquestions. The remainder of this chapter is organized as follows: Section 6.1 summarizes the research that answers to the research subquestions; Section 6.2 answers the dissertation question; and Section 6.3 discusses future work.

6.1 Research Subquestions

6.1.1 Answering RQ1. The first research thrust, presented in Chapter III, involved performing a user survey to determine scientific users' preferences. This research was the first to apply VQ metrics to scientific data and study the preference of scientific researchers. Overall, this work culminated in three new entropy-based metrics, and their combination was the best performing predictor of user preference at 68%.

6.1.2 Answering RQ2. The second research thrust, presented in Chapter IV, involved parallelizing VQ metrics using the VTK-m library [95] within the Ascent [71] infrastructure. VTK-m's use of parallel primitives guarantees portable performance because the parallel primitives can be mapped to any current or emerging parallel architecture for efficient usage, including exascale computers,

for efficient usage. By re-writing the VQ metrics using VTK-m, our metrics can make full use of parallel architectures. This research was the first time these VQ metrics have been parallelized and results of our performance study show they can be executed quickly for a large-scale simulation. Additionally, based on the performance study, it is reasonable to extrapolate the success of our approach to exascale simulations.

6.1.3 Answering RQ3. The third research thrust, presented in Chapter IV, utilized search algorithms and tested their ability to find a quality viewpoint quickly, first in a post-hoc setting and then in an in situ setting. In a post-hoc setting, we evaluated six search algorithms using individual cycles of large-scale scientific simulations and measured how quickly each search algorithm can find a quality viewpoint amongst all 10,000 views of each data set. This initial work proved that certain VQ metrics are easily searchable whereas other VQ metrics may produce search spaces that are lacking quality viewpoints and are thus harder to search. In an in situ setting, we evaluated the six search algorithms with a varying number of camera budgets and measured the tradeoff between the best VQ metric score and overall execution time. This thrust showed that, depending on the metric, a user only needs to consider 10-25 cameras in order to find a good viewpoint, whereas considering more than 25 cameras incurs execution costs that do not produce higher quality viewpoints in equal measure. In all, this research thrust proved that quality viewpoints can be found quickly and informs on best practices for how to find such viewpoints.

6.1.4 Answering RQ4. The final research thrust, presented in Chapter V, analyzed the behavior of camera viewpoints for time-varying data in order to determine how often to search for a new camera placement. This research

showed that, using a percentage based threshold, camera placement is relatively stable, and searching for a new camera placement happens infrequently, and thus amortizes the cost of searching for a new viewpoint over the lifespan of the simulation.

6.2 Dissertation Question

This dissertation aimed to answer the following question: **is automated in situ camera placement viable for large-scale simulations and what are the best practices?** This question was broken down into four research subquestions that in combination answer the dissertation questions. Answering the first subquestion showed that camera placement can be automated and produce renderings from viewpoints that users will want to see. This research performed a user survey to determine what viewpoints visualization experts and domain scientists prefer when visualizing scientific data. Further, it resulted in the creation of three new VQ metrics that, when added together, is the best predictor of user preference at 68%. Answering the second subquestion showed that VQ metrics can be performed in situ efficiently for large-scale simulations. This research rewrote the VQ metrics using VTK-m, allowing them to make effective use of shared-memory parallelism, shown by the large-scale performance study and the efficient use of OpenMP on the backend. Answering the third subquestion showed that quality viewpoints, as determined by VQ metric score, can be found quickly without having to compute an encumbering number of camera placements. Utilizing multiple search algorithms and multiple camera budgets, this research showed that searching the data space diagonally or randomly can produce a quality viewpoint with a camera budget that should not exceed 20. Answering the fourth subquestion showed that camera placement over time fairly stable and

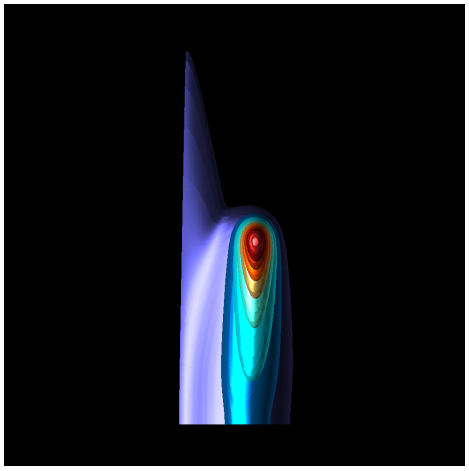
using a trigger-based search routine limits the frequency of performing a search to instances of change within the simulation. This research resulted in a threshold trigger based on the percentage of the best VQ metric score, which, based on the chosen percentage, can be used to limit the frequency of camera searches. In all, this research has shown that automatic camera placement is viable for in situ visualization and has informed on best practices for running at scale.

6.3 Future Work

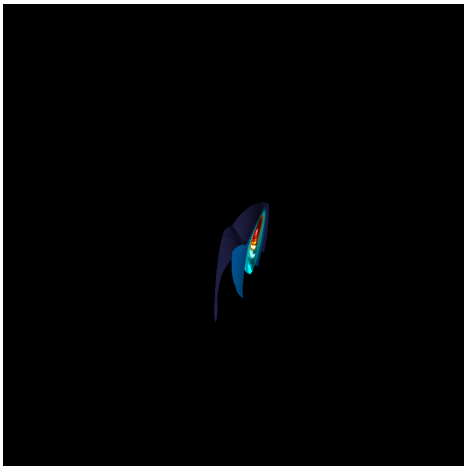
Future work for this research is numerous. Certainly there is more work needed in developing new VQ metrics, in addition to formulating a combination of VQ metrics, that better align with users preferences. Additionally, there should be further user surveys to enlarge the corpus of user preference that has been established with this work. Another important direction is types of data, and we feel this work should be expanded to include other types of scientific data other than scalar data, such as flow or volume data. The type of output of this work should also be explored. Instead of outputting a single image, it could output several of the best images. Or when the optimal viewpoint changes, outputting a number of transitionary images or video that follows a an informative path from the old optimal viewpoint to the new optimal viewpoint. In a final area of future work, there are many additional aspects of scientific visualization and analysis that are traditionally done in a post-hoc setting with a HITL and that require automate those decisions for in situ implementation.

APPENDIX

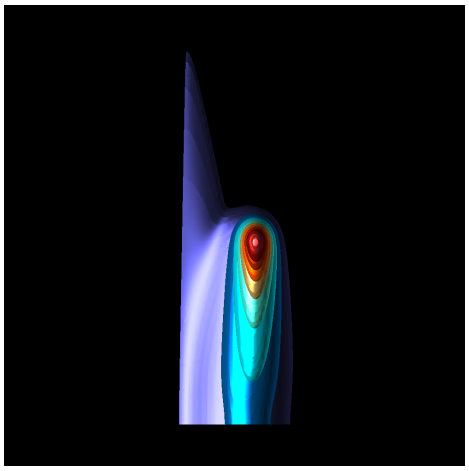
BEST AND WORST IMAGES FROM EVALUATION IN CHAPTER II



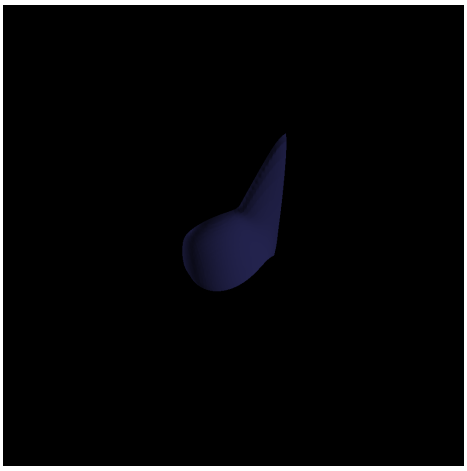
(a) VQ_1 Best



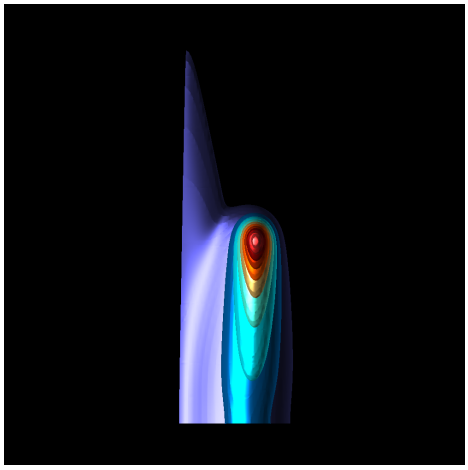
(b) VQ_1 Worst



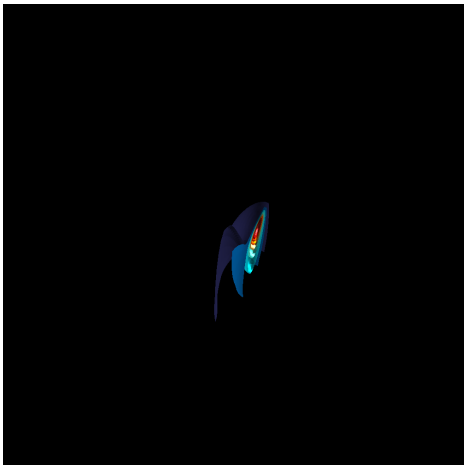
(c) VQ_2 Best



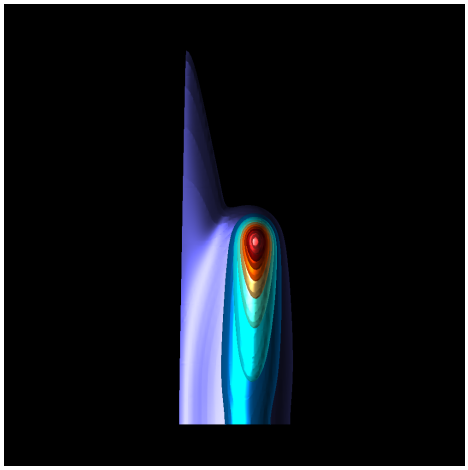
(d) VQ_2 Worst



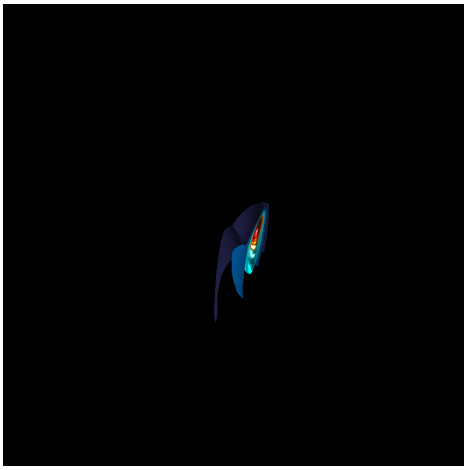
(e) VQ_3 Best



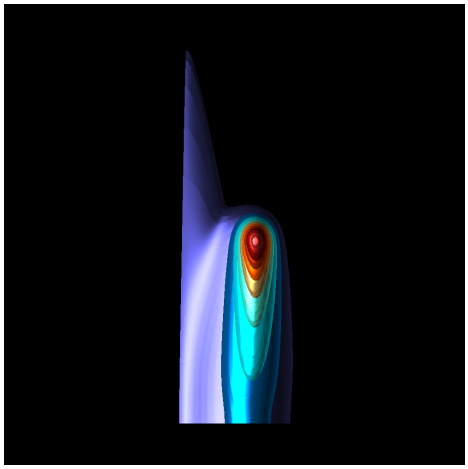
(f) VQ_3 Worst



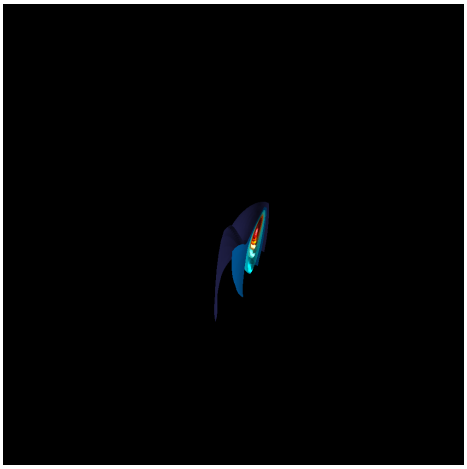
(g) VQ_4 Best



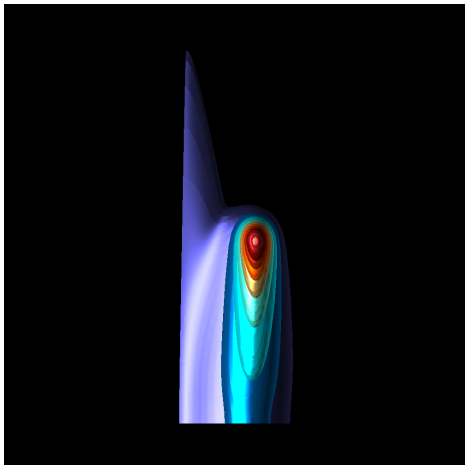
(h) VQ_4 Worst



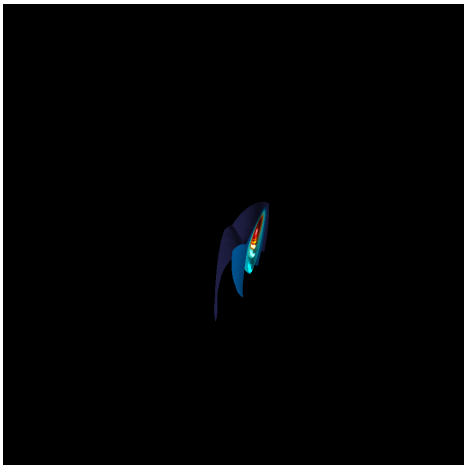
(i) VQ_5 Best



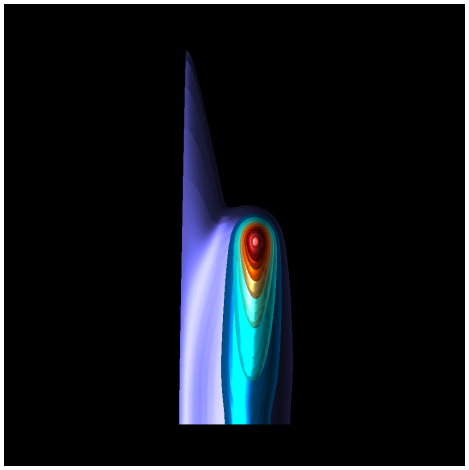
(j) VQ_5 Worst



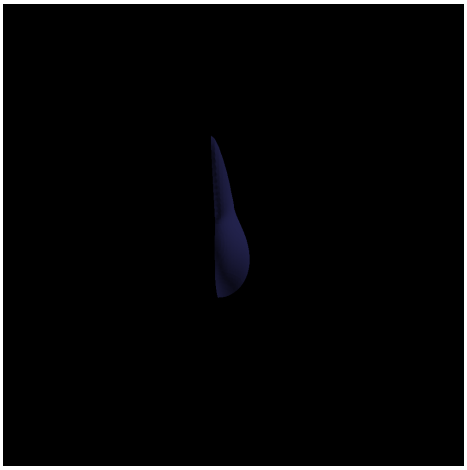
(k) VQ_6 Best



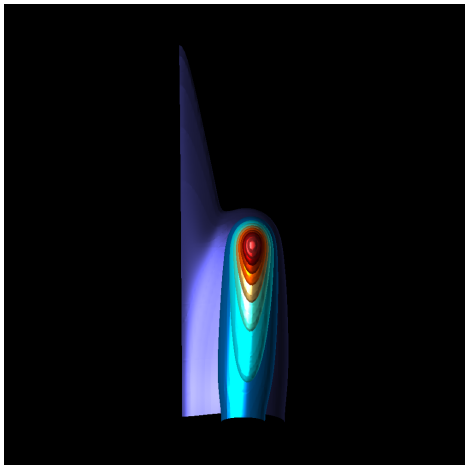
(l) VQ_6 Worst



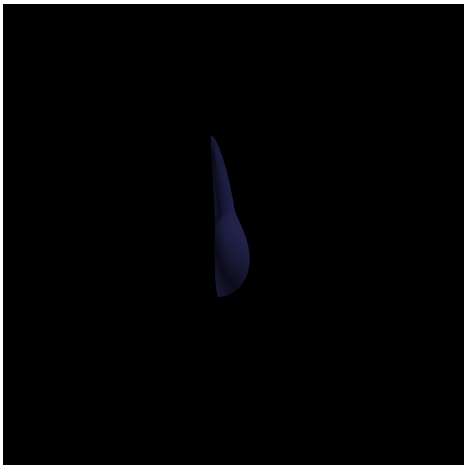
(m) VQ_7 Best



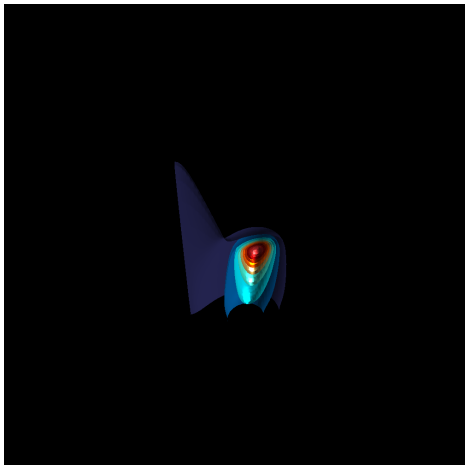
(n) VQ_7 Worst



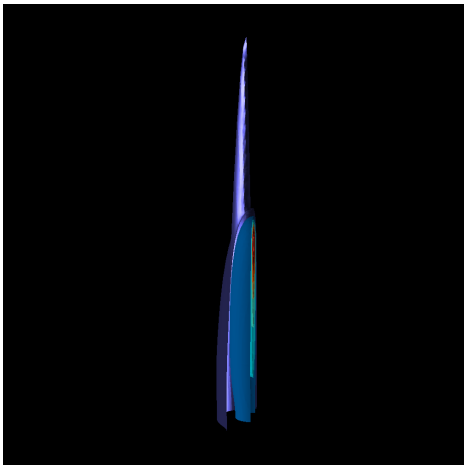
(o) VQ_{10} Best



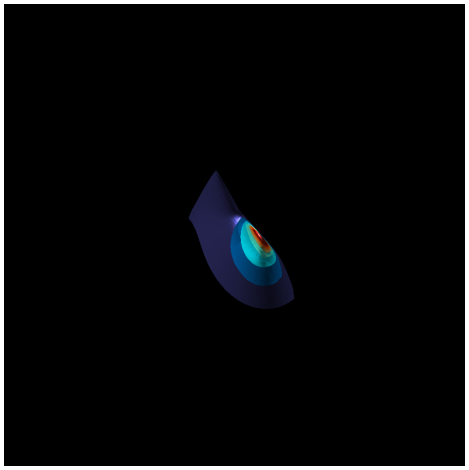
(p) VQ_{10} Worst



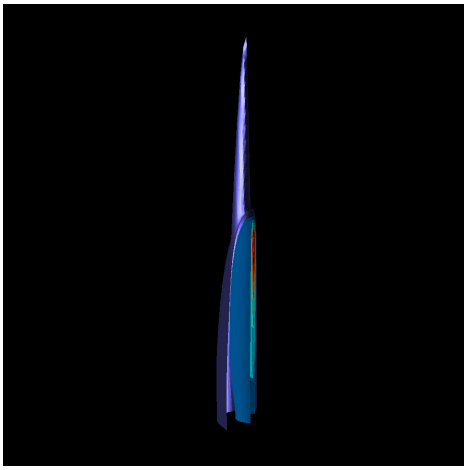
(q) VQ_{11} Best



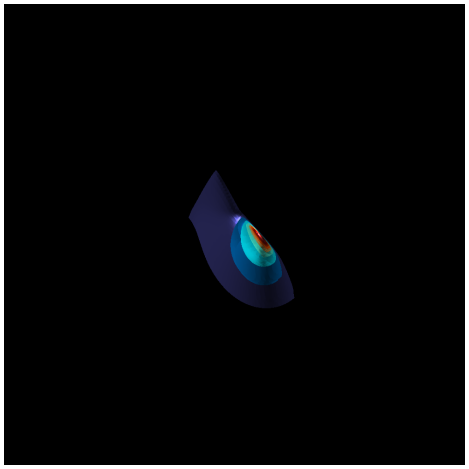
(r) VQ_{11} Worst



(s) VQ_{12} Best

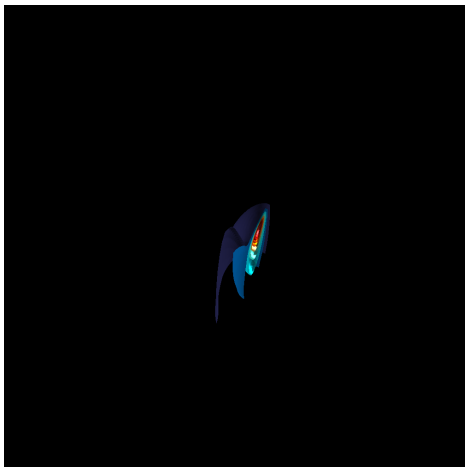


(t) VQ_{12} Worst

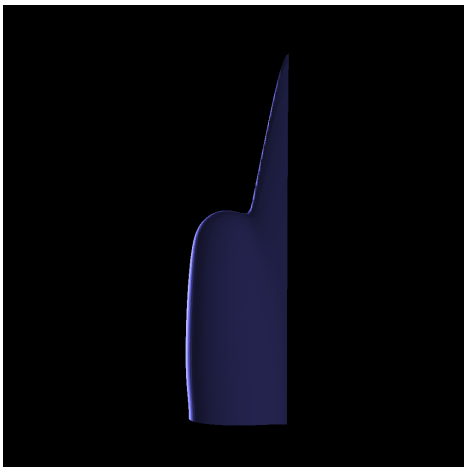


(u) VQ_{13} Best

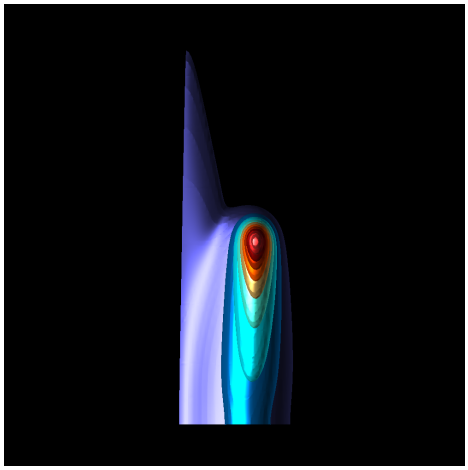
(v) VQ_{13} Worst



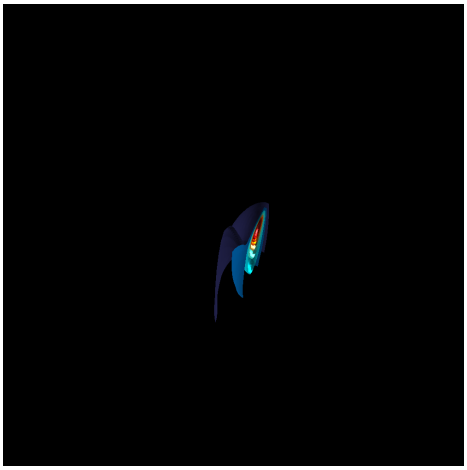
(w) VQ_{14} Best



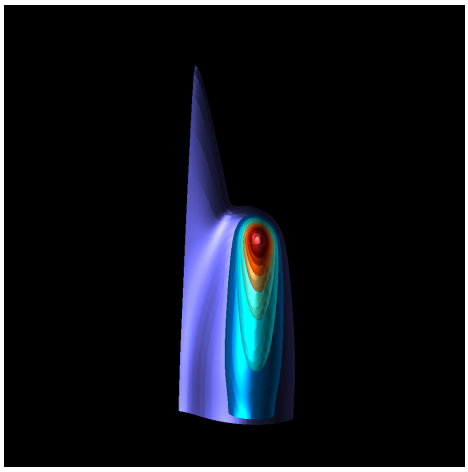
(x) VQ_{14} Worst



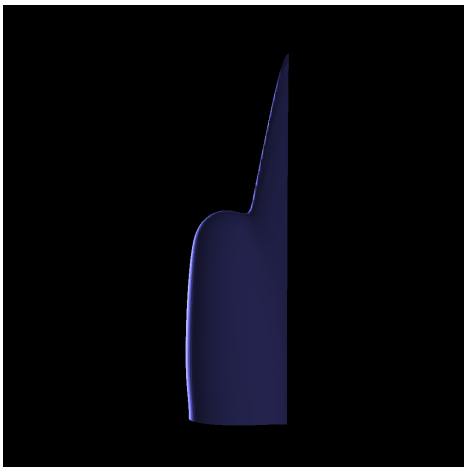
(y) VQ_{15} Best



(z) VQ_{15} Worst

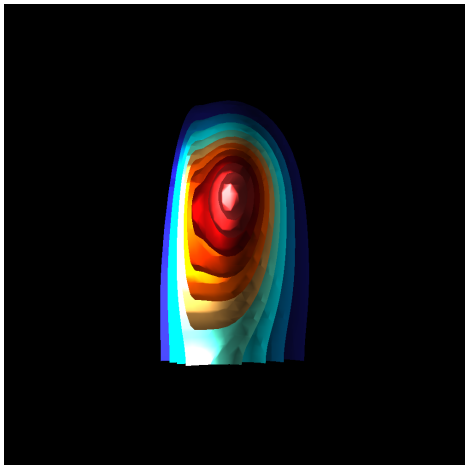


(aa) VQ_{16} Best

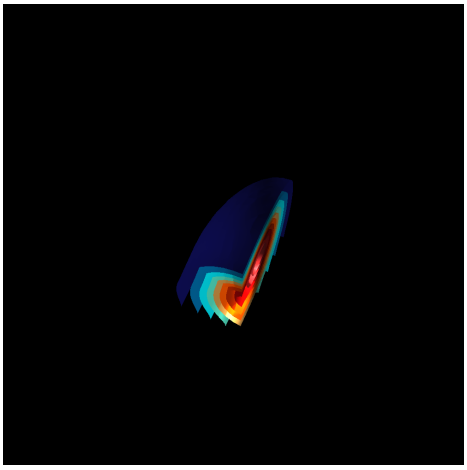


(ab) VQ_{16} Worst

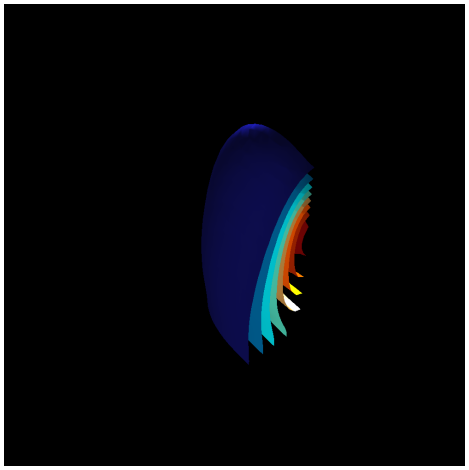
Figure A.43. The best and worst images for the implemented metrics on the ExaAM #1 timestep with 21,255 triangles.



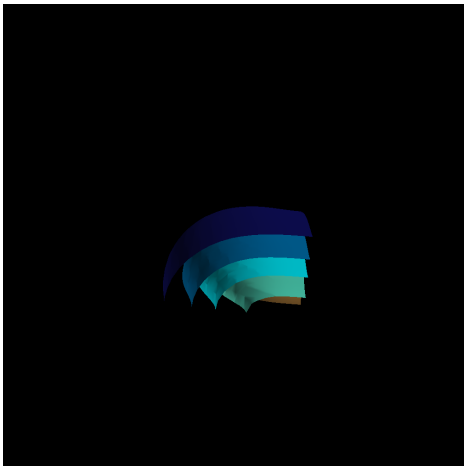
(a) VQ_1 Best



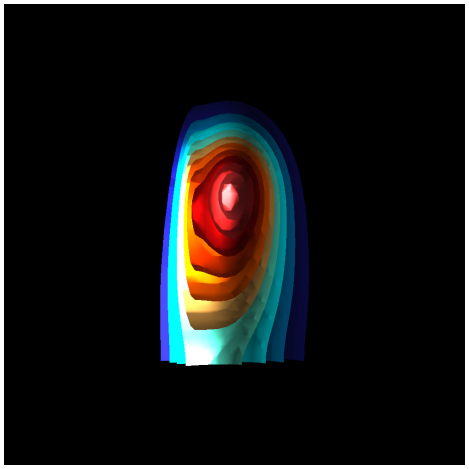
(b) VQ_1 Worst



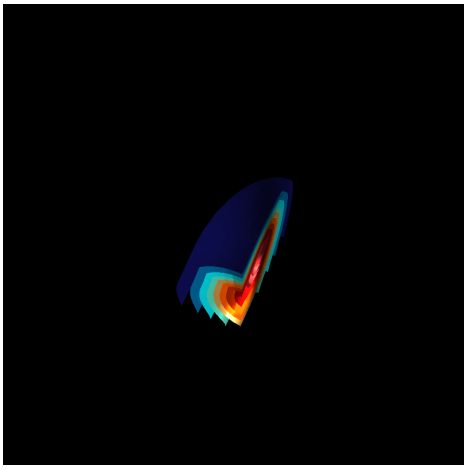
(c) VQ_2 Best



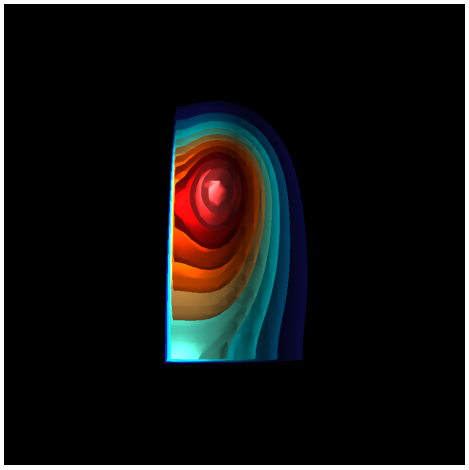
(d) VQ_2 Worst



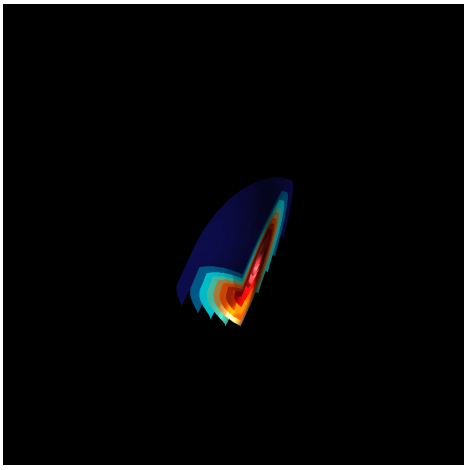
(e) VQ_3 Best



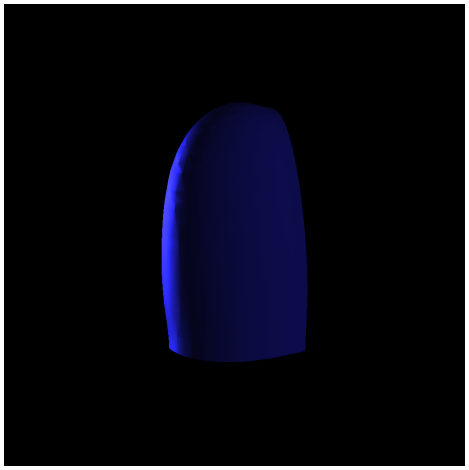
(f) VQ_3 Worst



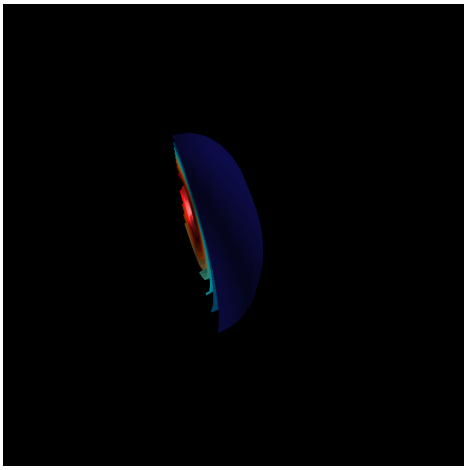
(g) VQ_4 Best



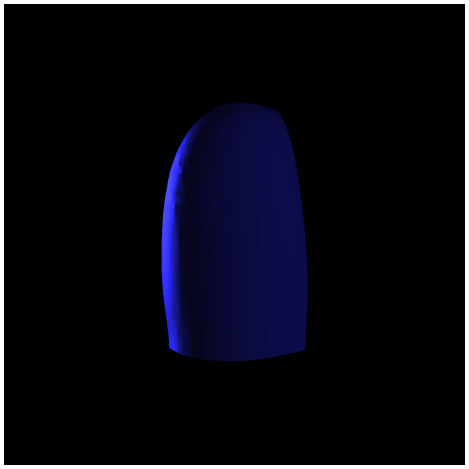
(h) VQ_4 Worst



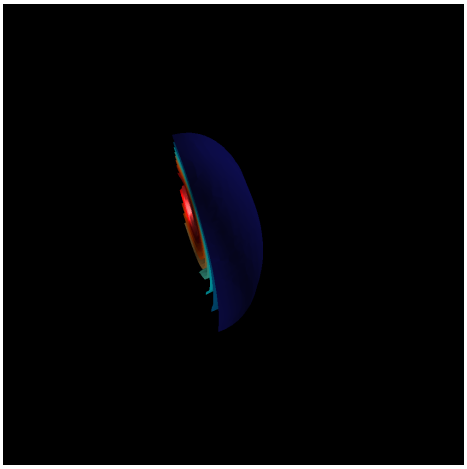
(i) VQ_5 Best



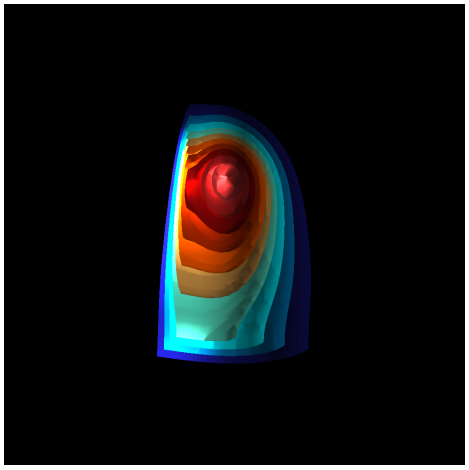
(j) VQ_5 Worst



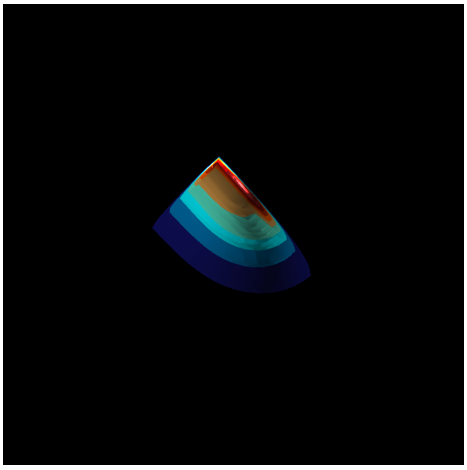
(k) VQ_6 Best



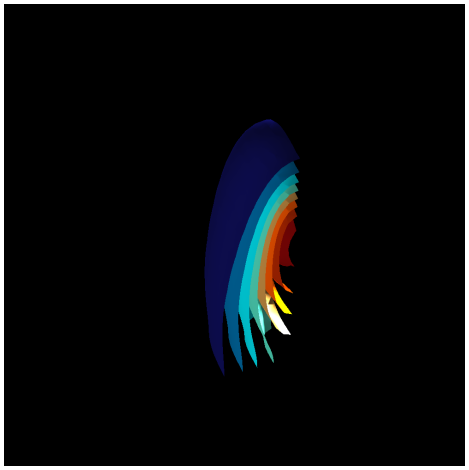
(l) VQ_6 Worst



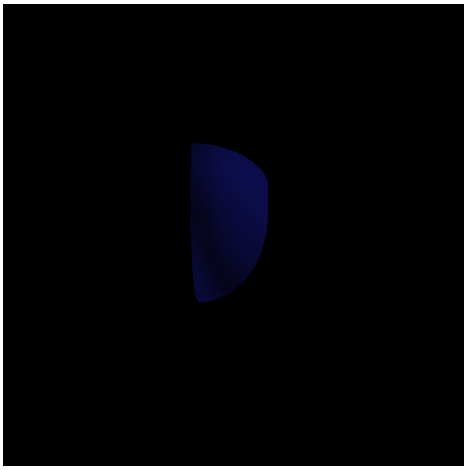
(m) VQ_7 Best



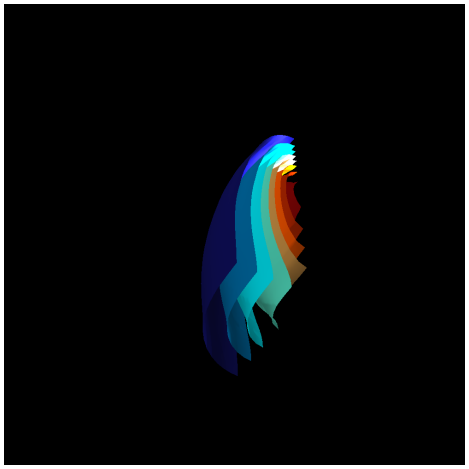
(n) VQ_7 Worst



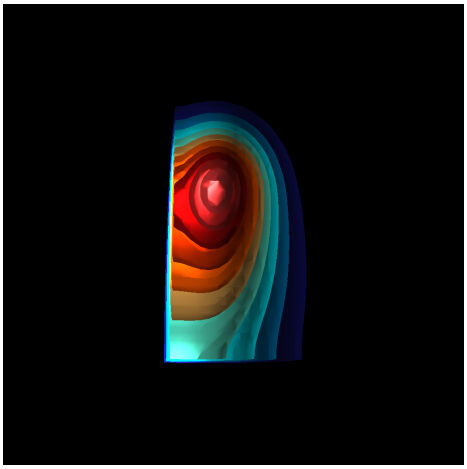
(o) VQ_{10} Best



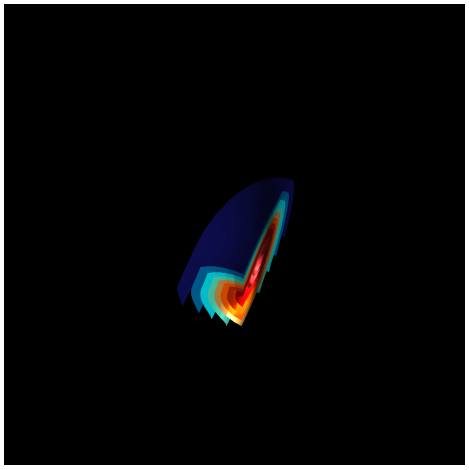
(p) VQ_{10} Worst



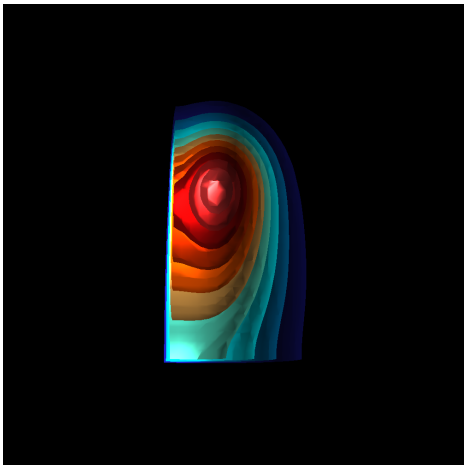
(q) VQ_{11} Best



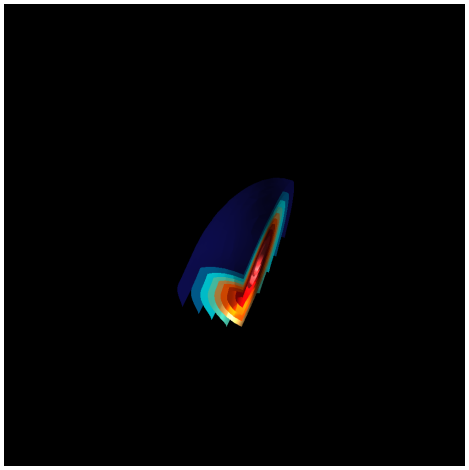
(r) VQ_{11} Worst



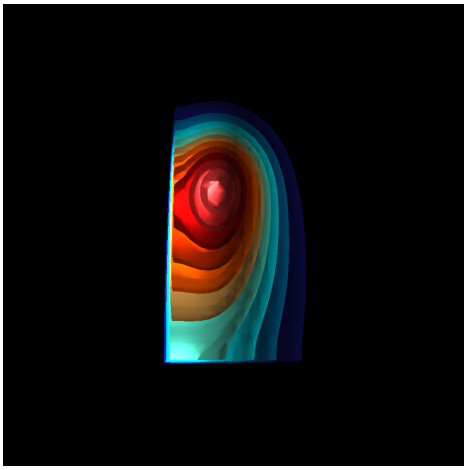
(s) VQ_{12} Best



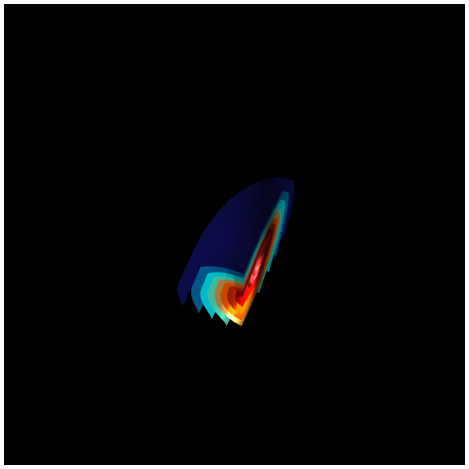
(t) VQ_{12} Worst



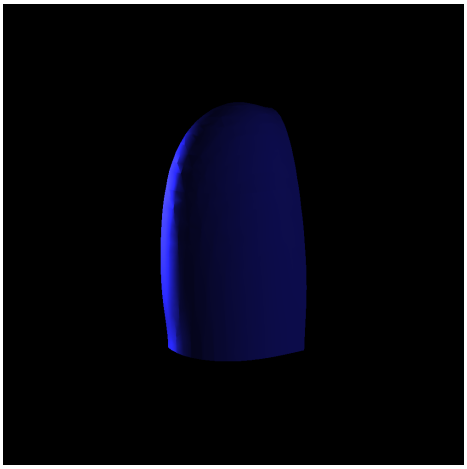
(u) VQ_{13} Best



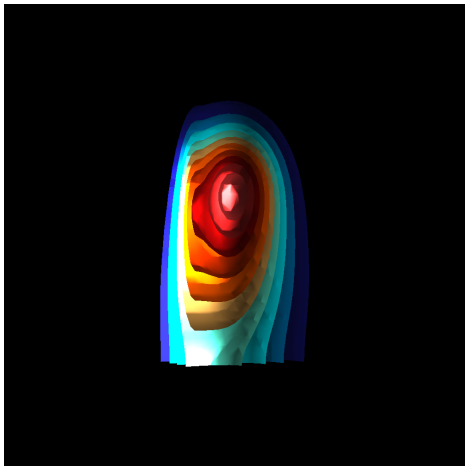
(v) VQ_{13} Worst



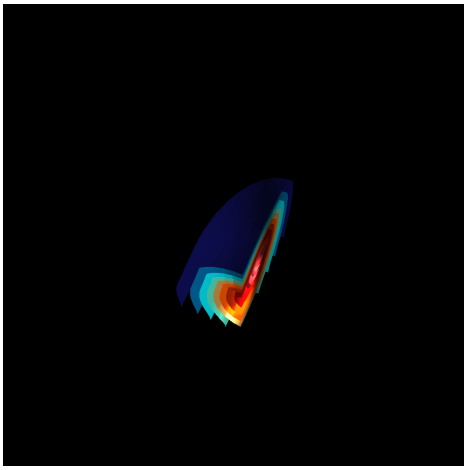
(w) VQ_{14} Best



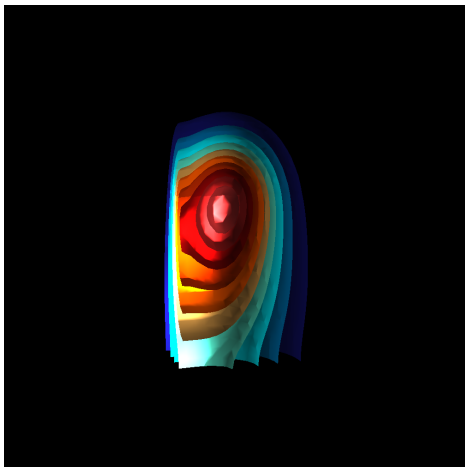
(x) VQ_{14} Worst



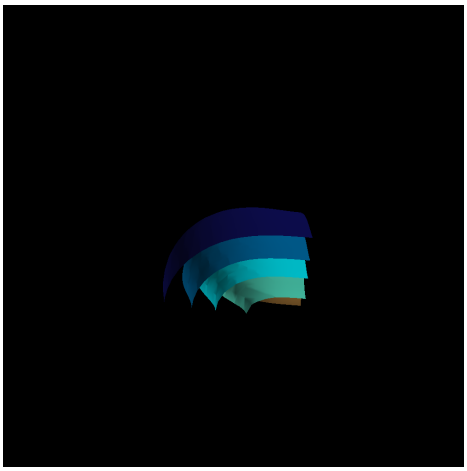
(y) VQ_{15} Best



(z) VQ_{15} Worst

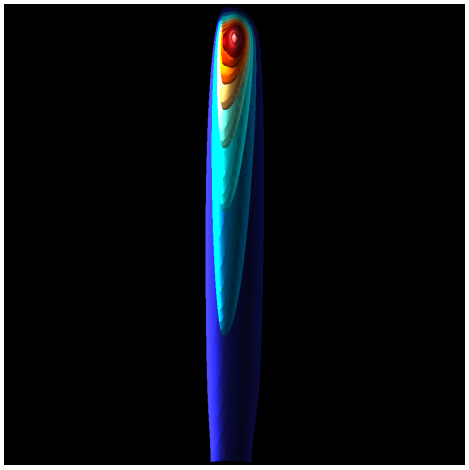


(aa) VQ_{16} Best

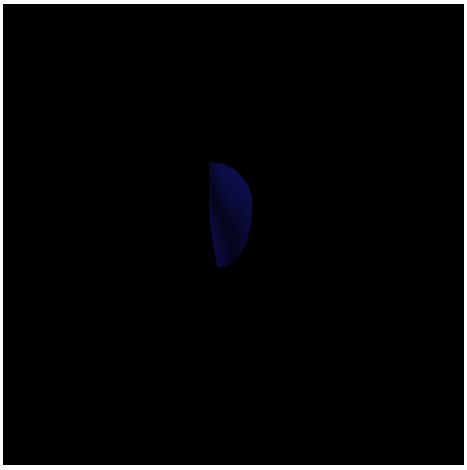


(ab) VQ_{16} Worst

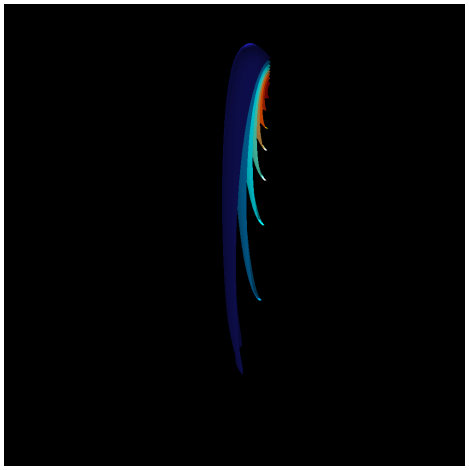
Figure A.44. The best and worst images for the implemented metrics on the ExaAM #2 timestep with 6,474 triangles.



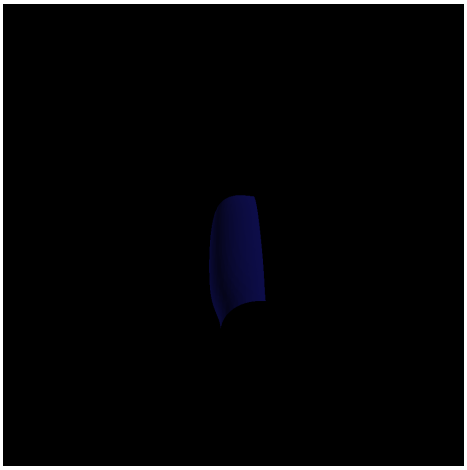
(a) VQ_1 Best



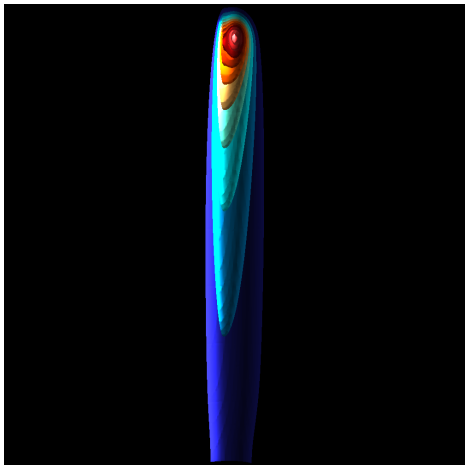
(b) VQ_1 Worst



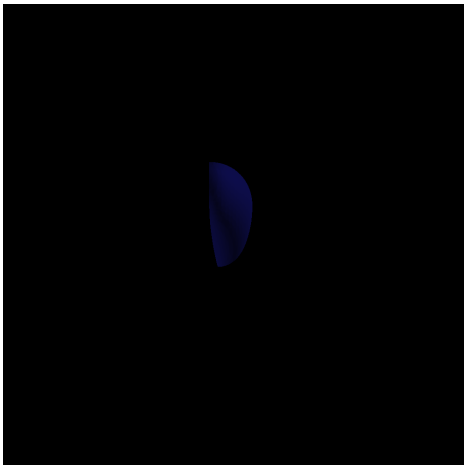
(c) VQ_2 Best



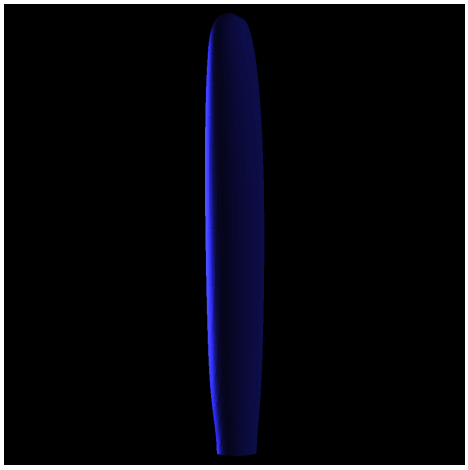
(d) VQ_2 Worst



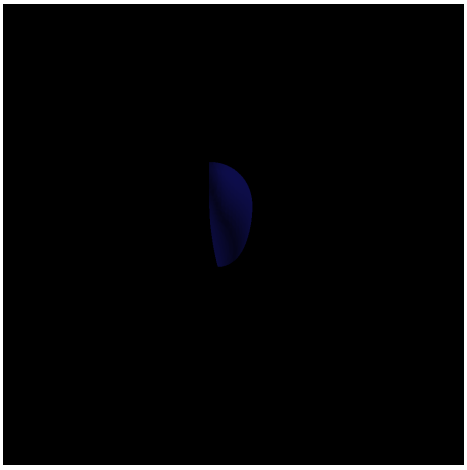
(e) VQ_3 Best



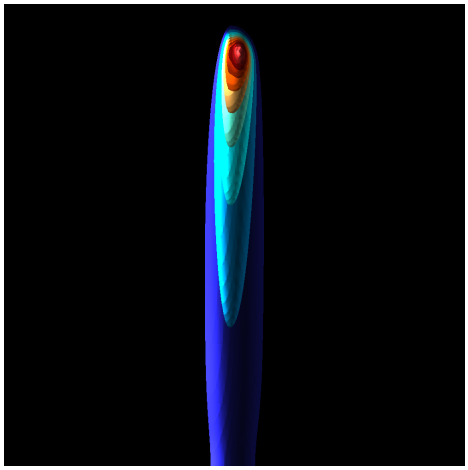
(f) VQ_3 Worst



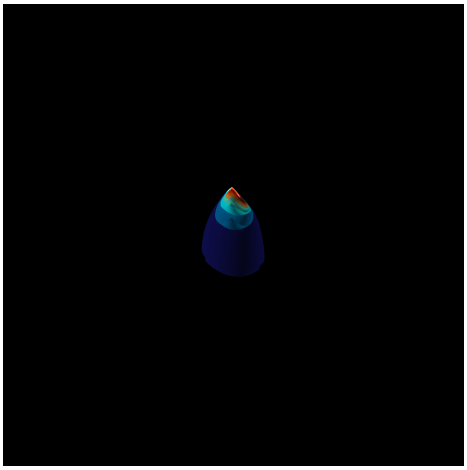
(g) VQ_4 Best



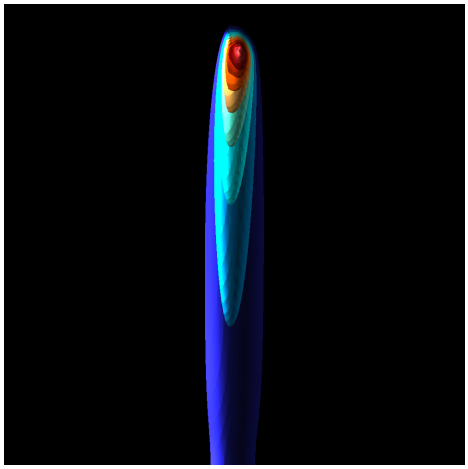
(h) VQ_4 Worst



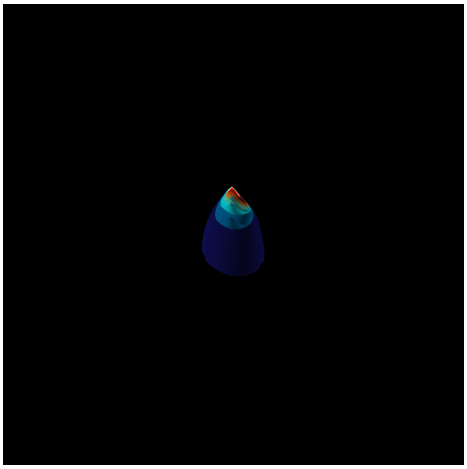
(i) VQ_5 Best



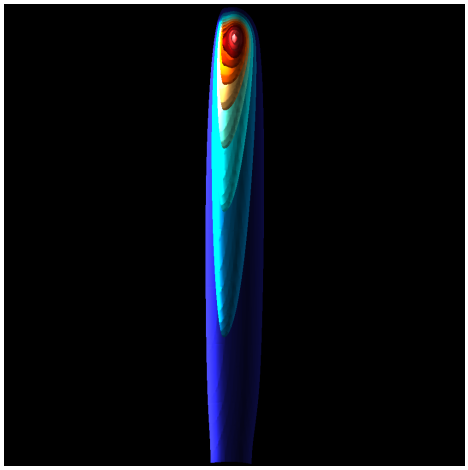
(j) VQ_5 Worst



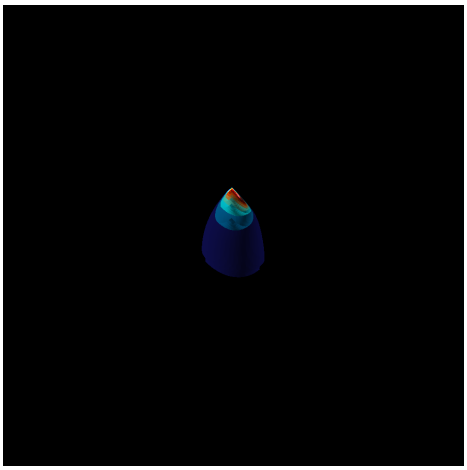
(k) VQ_6 Best



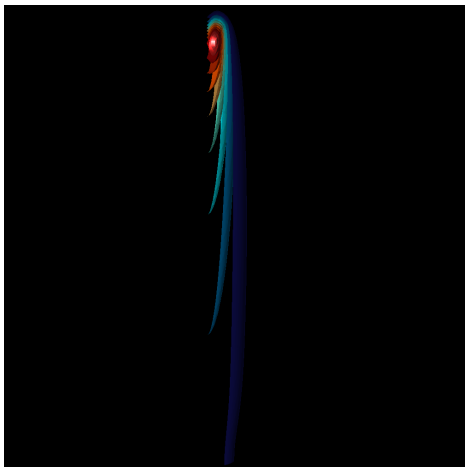
(l) VQ_6 Worst



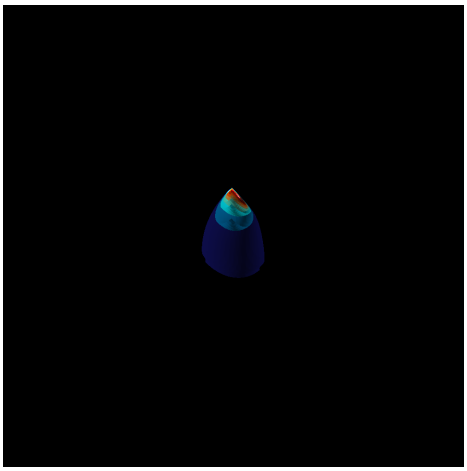
(m) [ExaAM 3 VQ_7 Best]VQ_7 Best



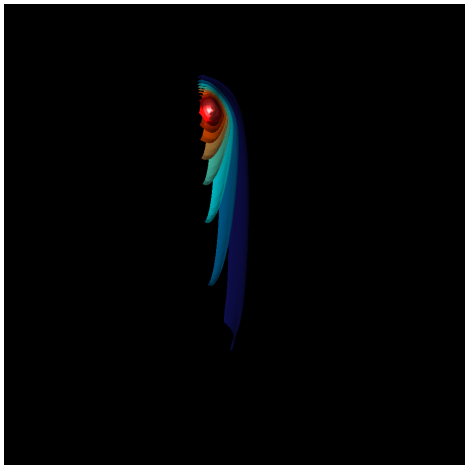
(n) VQ_7 Worst



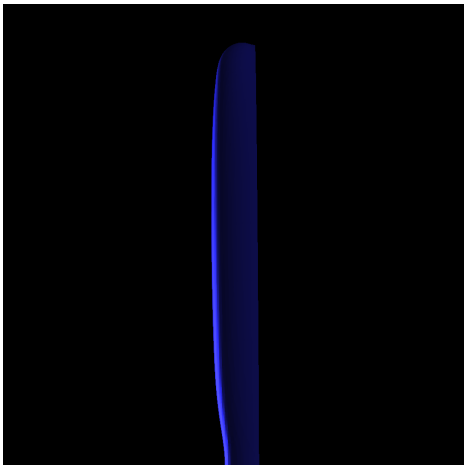
(o) VQ_{10} Best



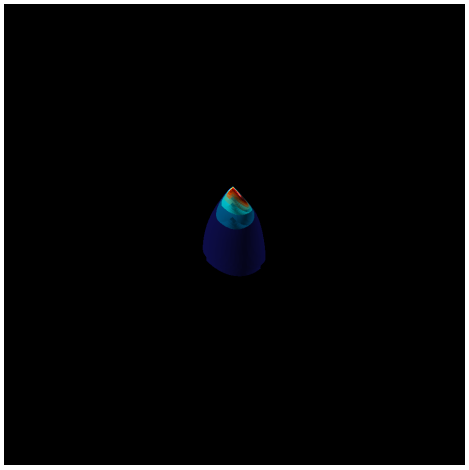
(p) VQ_{10} Worst



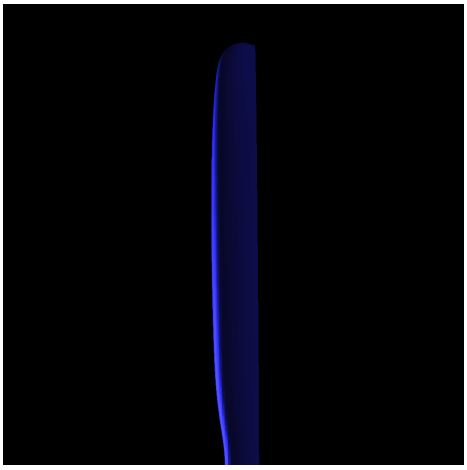
(q) VQ_{11} Best



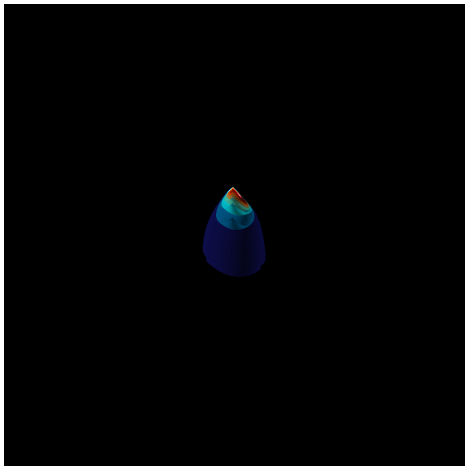
(r) VQ_{11} Worst



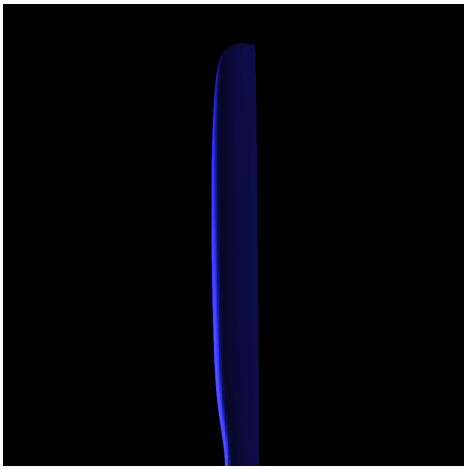
(s) VQ_{12} Best



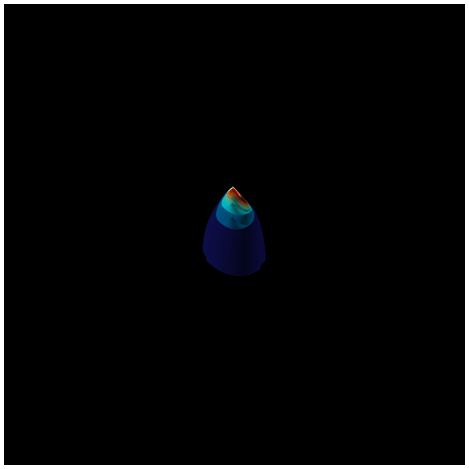
(t) VQ_{12} Worst



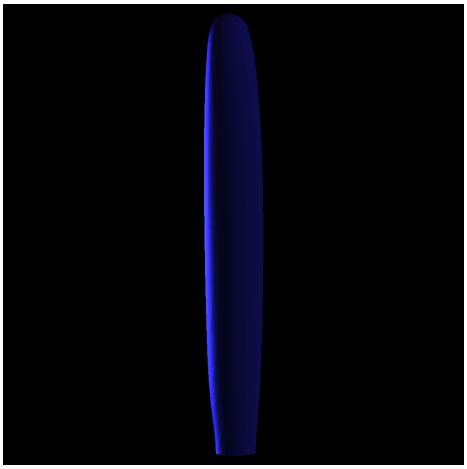
(u) VQ_{13} Best



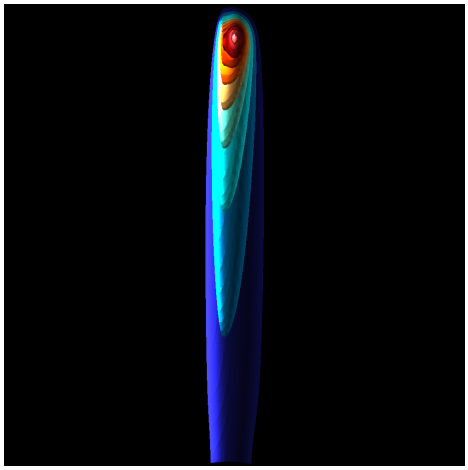
(v) VQ_{13} Worst



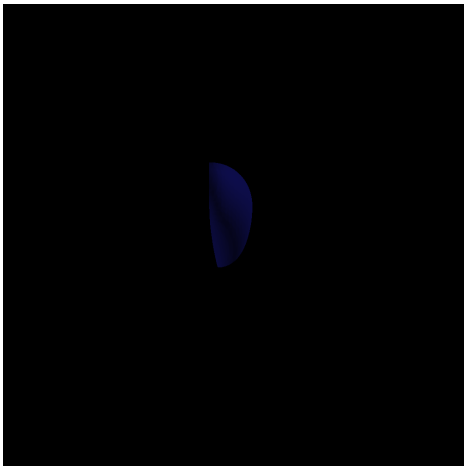
(w) VQ_{14} Best



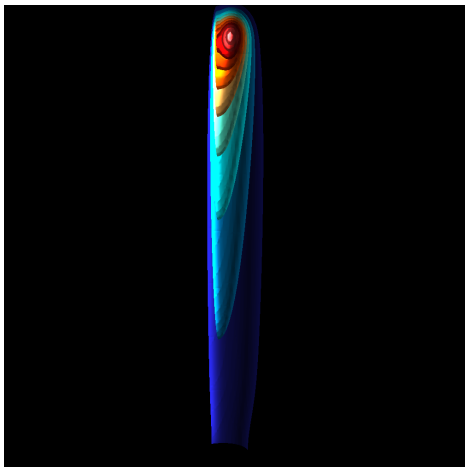
(x) VQ_{14} Worst



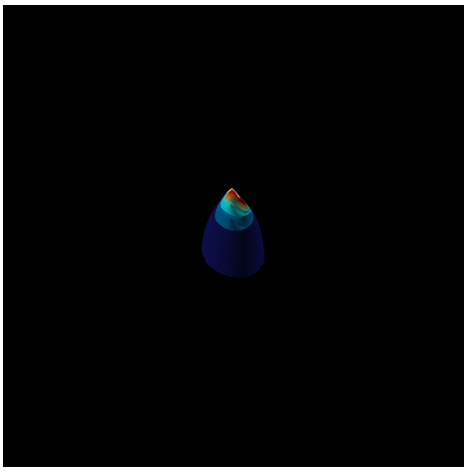
(y) VQ_{15} Best



(z) VQ_{15} Worst

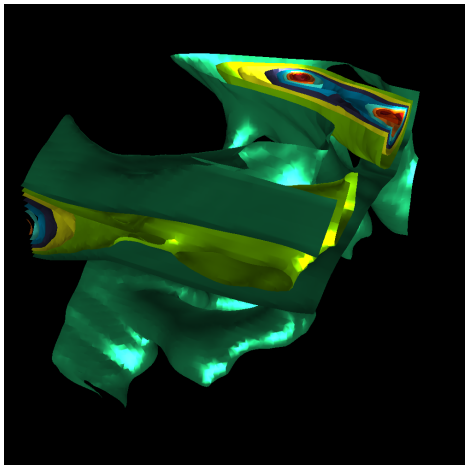


(aa) VQ_{16} Best

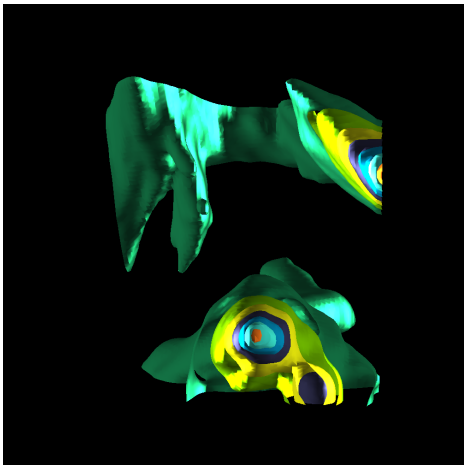


(ab) VQ_{16} Worst

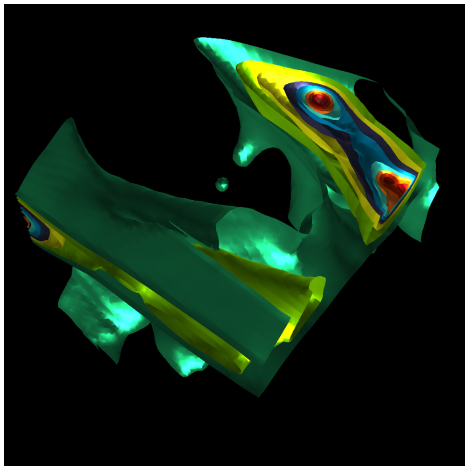
Figure A.45. The best and worst images for the implemented metrics on the ExaAM #3 timestep with 18,473 triangles.



(a) VQ_1 Best

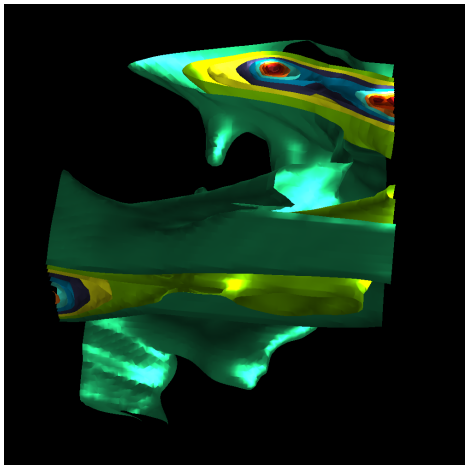


(b) VQ_1 Worst

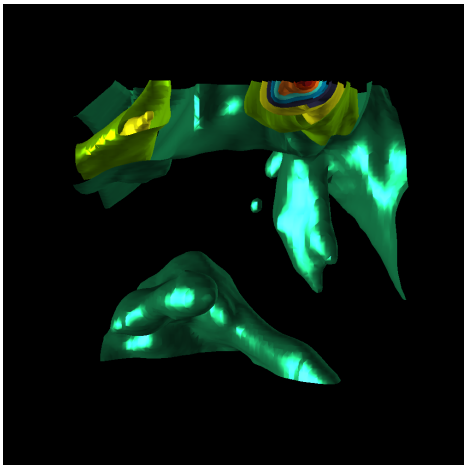


(c) VQ_2 Best

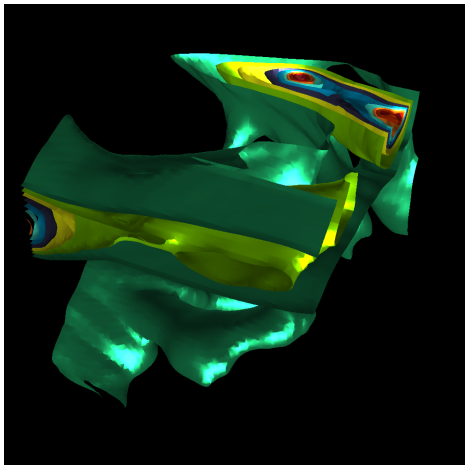
(d) VQ_2 Worst



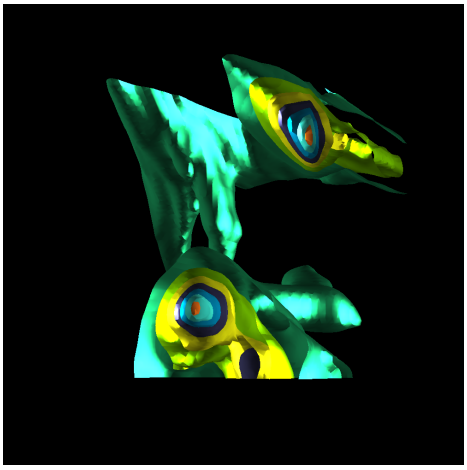
(e) VQ_3 Best



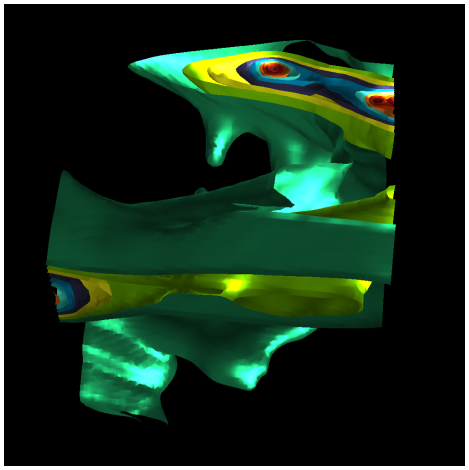
(f) VQ_3 Worst



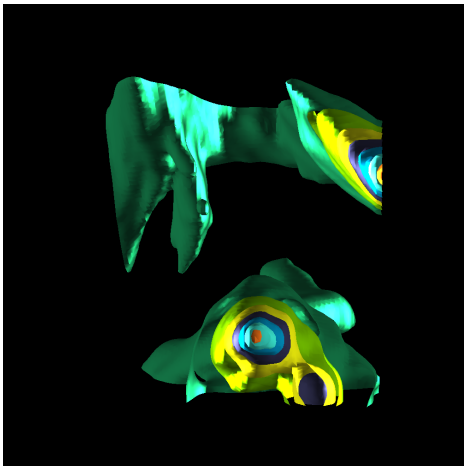
(g) VQ_4 Best



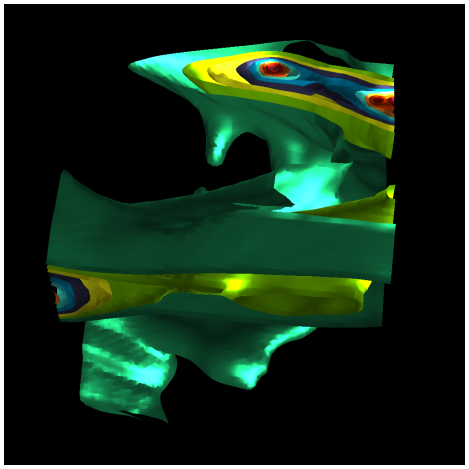
(h) VQ_4 Worst



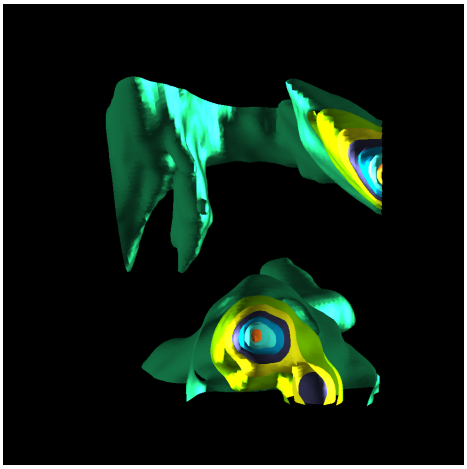
(i) VQ_5 Best



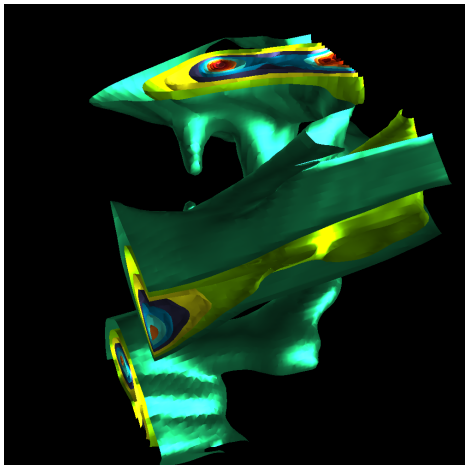
(j) VQ_5 Worst



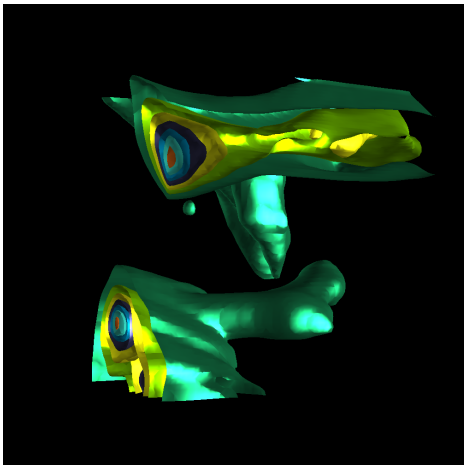
(k) VQ_6 Best



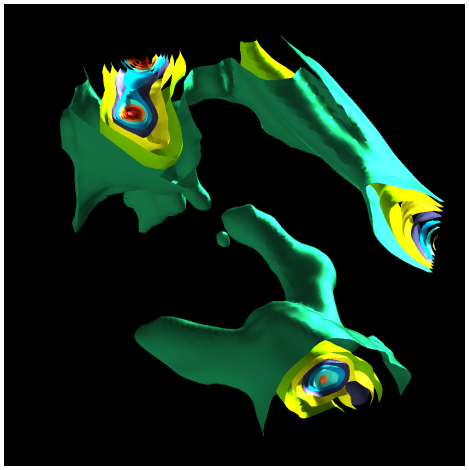
(l) VQ_6 Worst



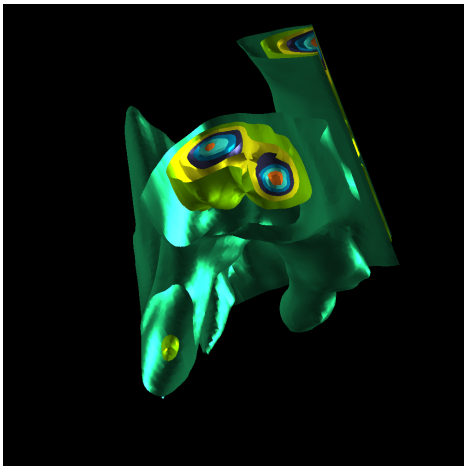
(m) VQ_7 Best



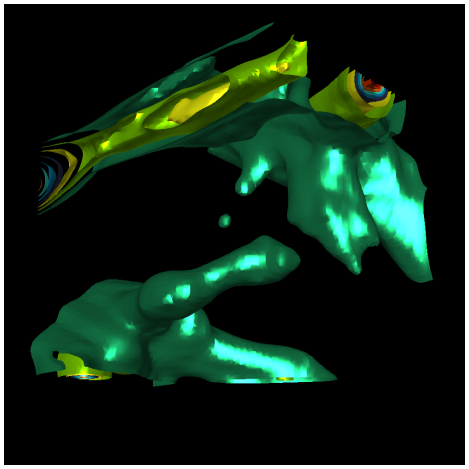
(n) VQ_7 Worst



(o) VQ_{10} Best

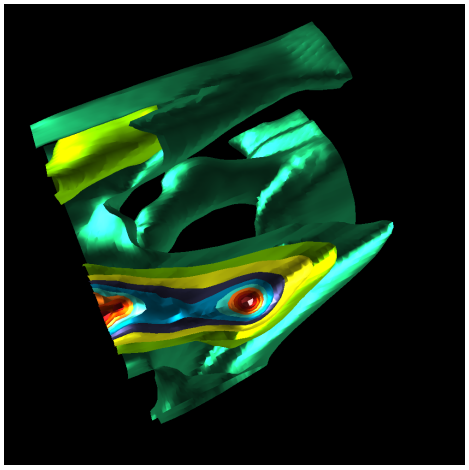


(p) VQ_{10} Worst



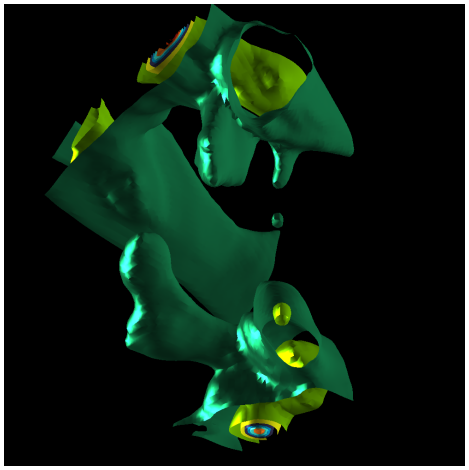
(q) VQ_{11} Best

(r) VQ_{11} Worst



(s) VQ_{12} Best

(t) VQ_{12} Worst

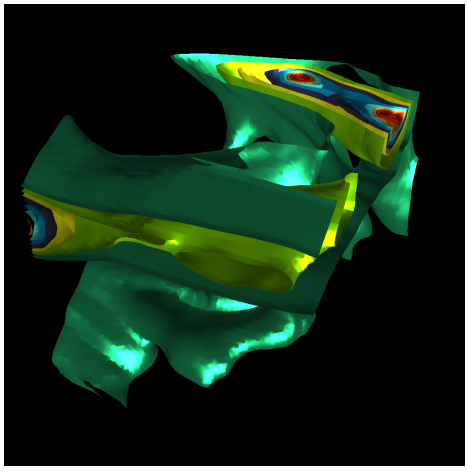


(u) VQ_{13} Best

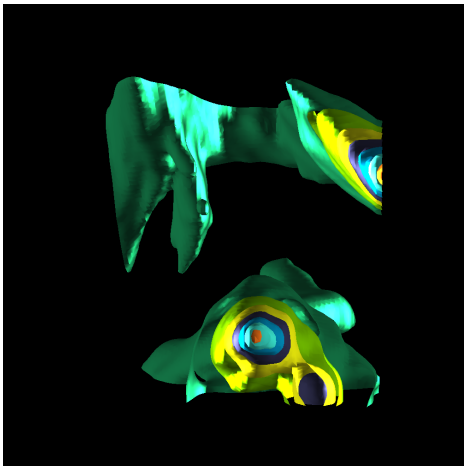
(v) VQ_{13} Worst

(w) VQ_{14} Best

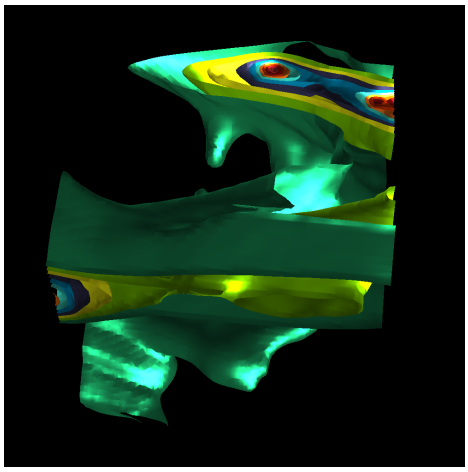
(x) VQ_{14} Worst



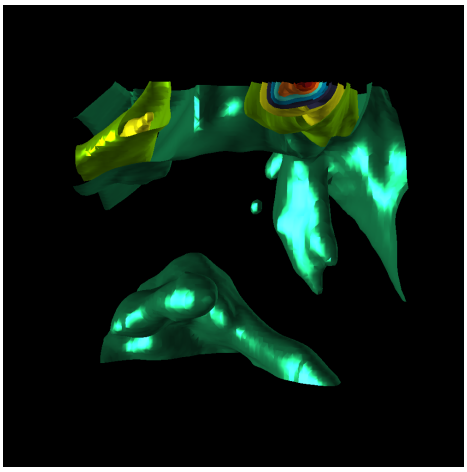
(y) VQ_{15} Best



(z) VQ_{15} Worst

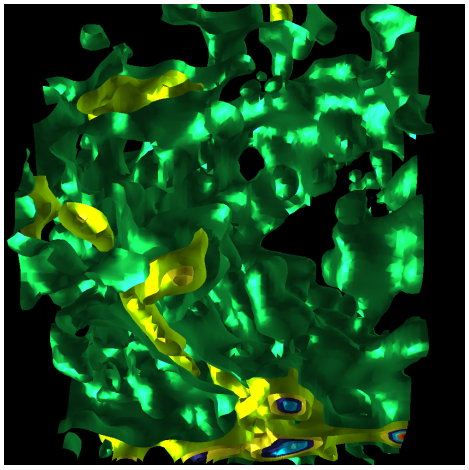


(aa) VQ_{16} Best

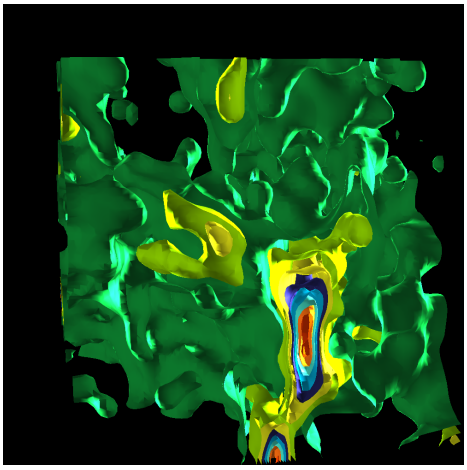


(ab) VQ_{16} Worst

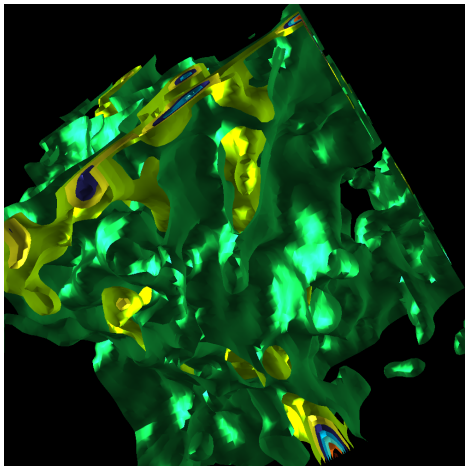
Figure A.46. The best and worst images for the implemented metrics on the ExaSky #1 timestep with 55,544 triangles.



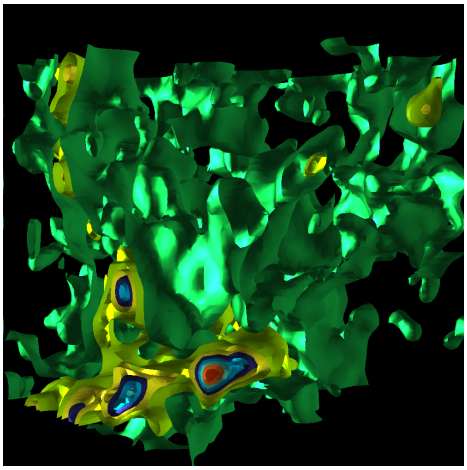
(a) VQ_1 Best



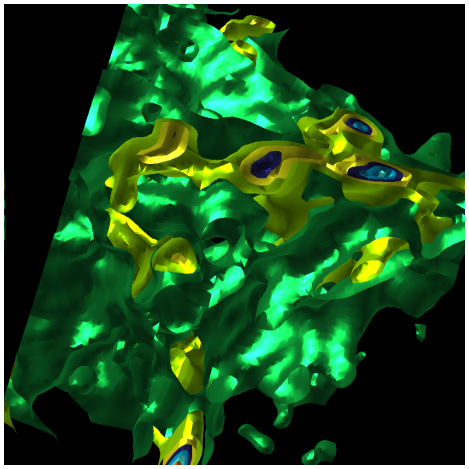
(b) VQ_1 Worst



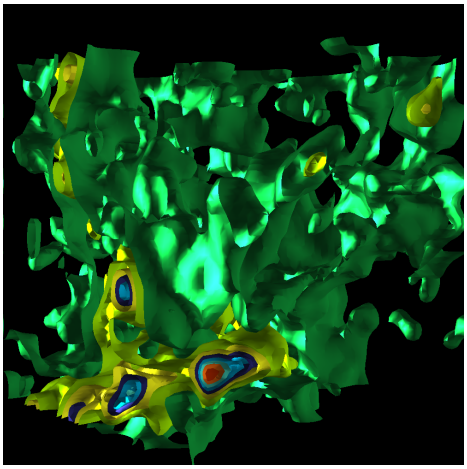
(c) VQ_2 Best



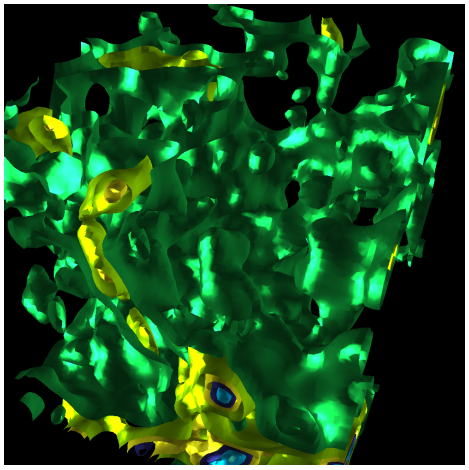
(d) VQ_2 Worst



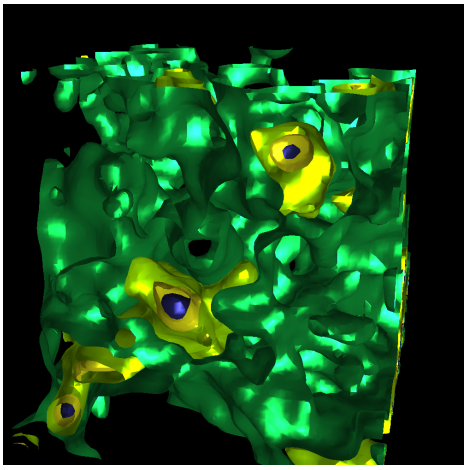
(e) VQ_3 Best



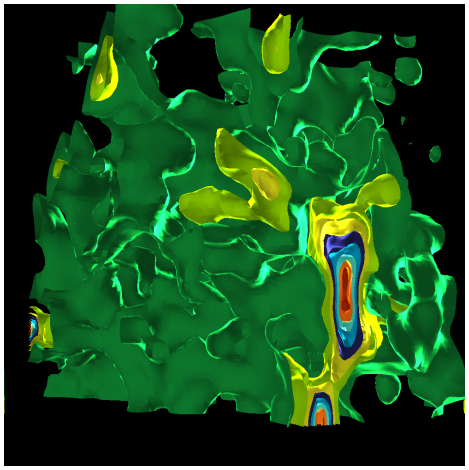
(f) VQ_3 Worst



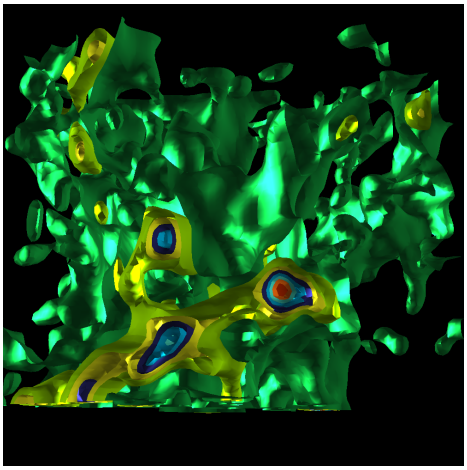
(g) VQ_4 Best



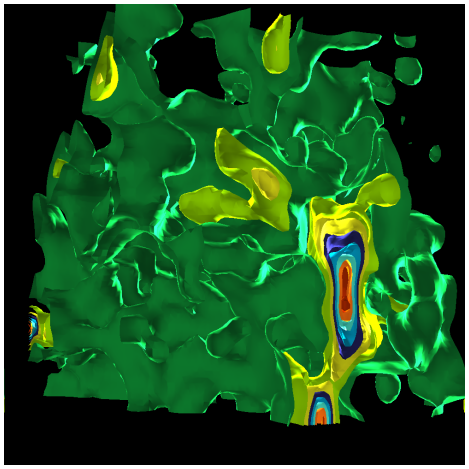
(h) VQ_4 Worst



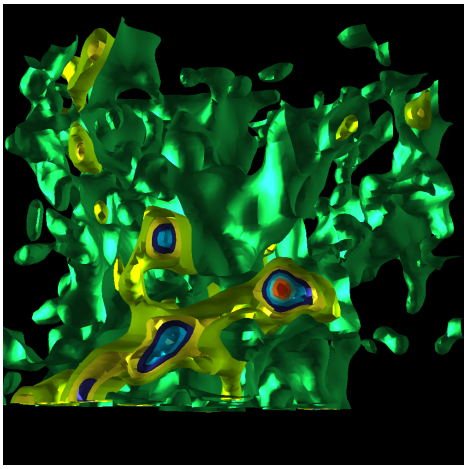
(i) VQ_5 Best



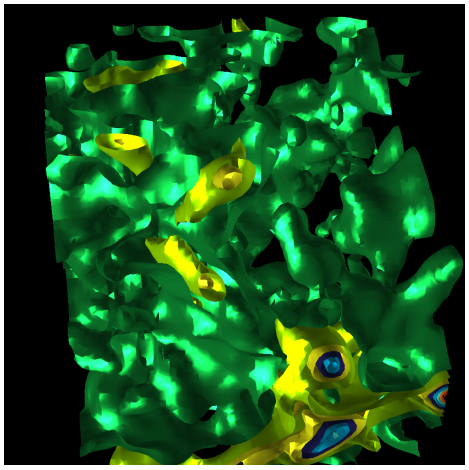
(j) VQ_5 Worst



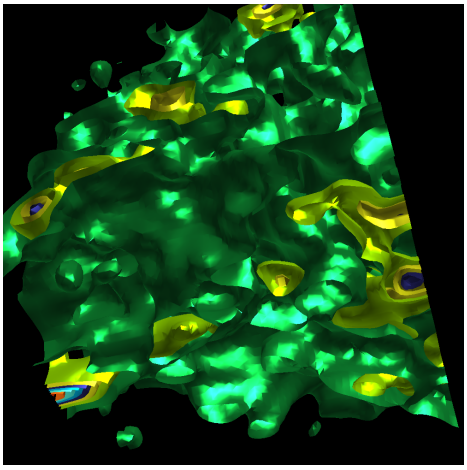
(k) VQ_6 Best



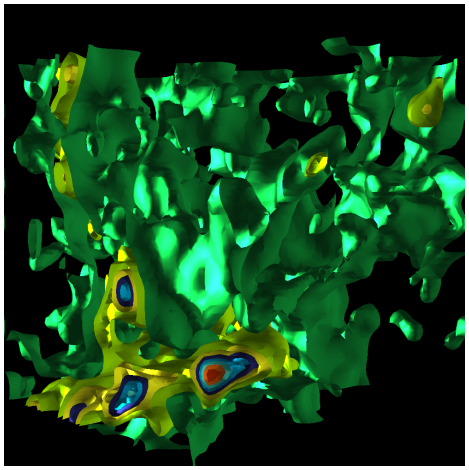
(l) VQ_6 Worst



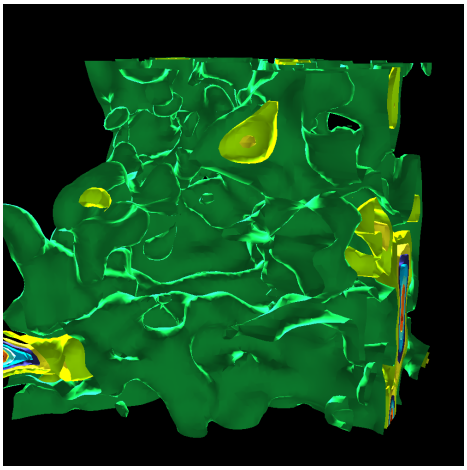
(m) VQ_7 Best



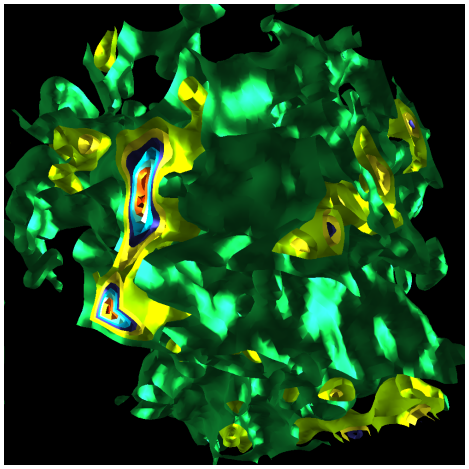
(n) VQ_7 Worst



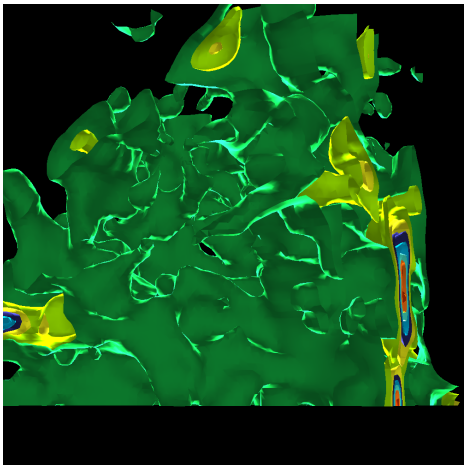
(o) VQ_{10} Best



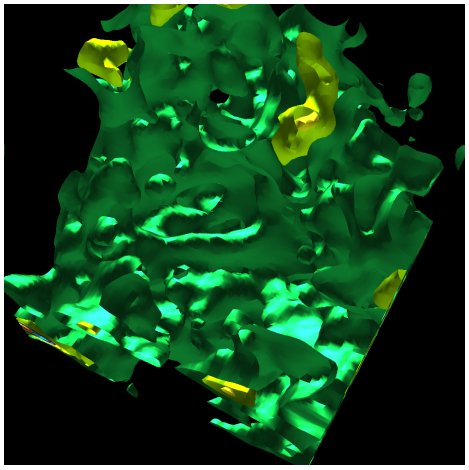
(p) VQ_{10} Worst



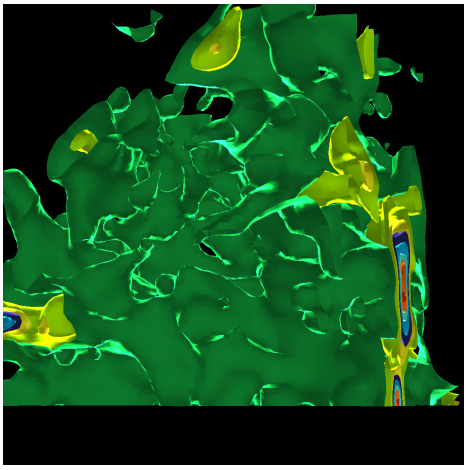
(q) VQ_{11} Best



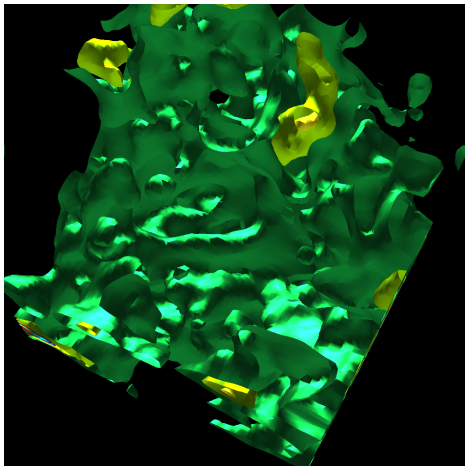
(r) VQ_{11} Worst



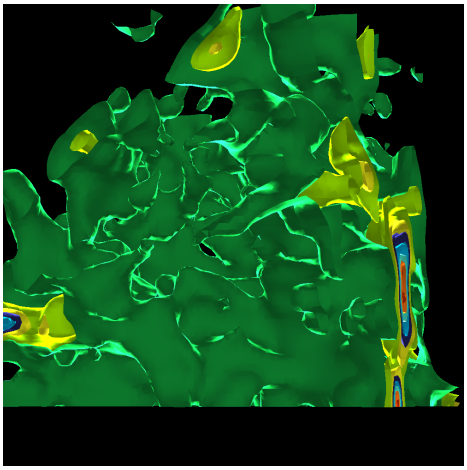
(s) VQ_{12} Best



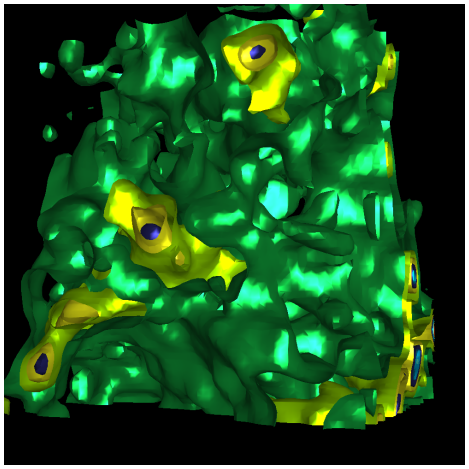
(t) VQ_{12} Worst



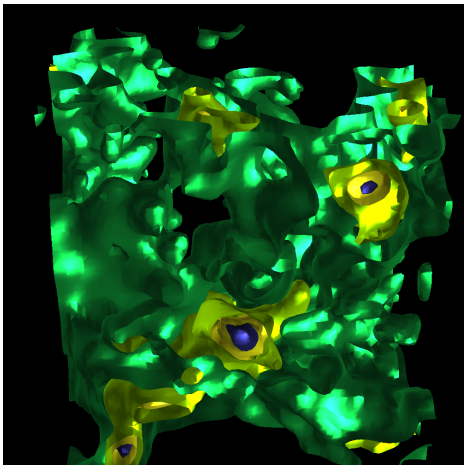
(u) VQ_{13} Best



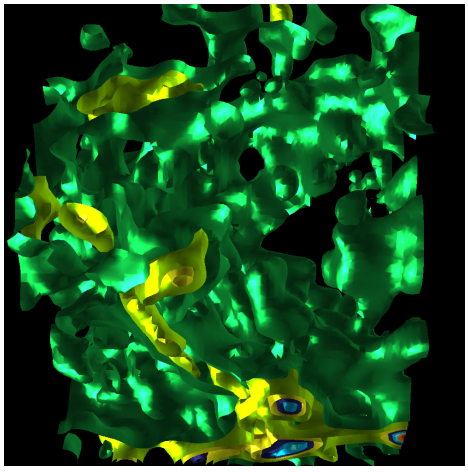
(v) VQ_{13} Worst



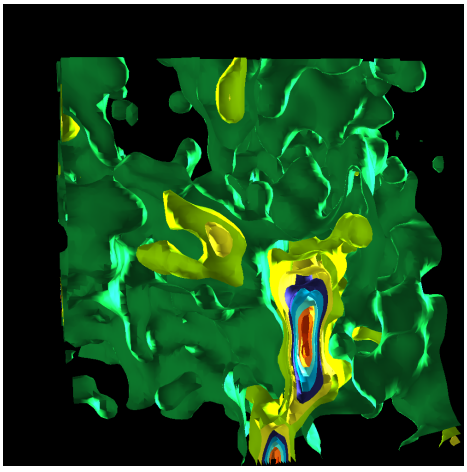
(w) VQ_{14} Best



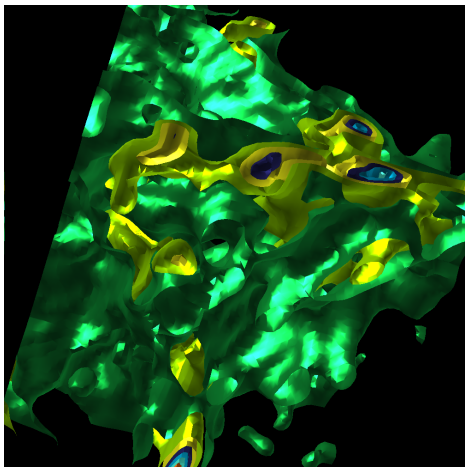
(x) VQ_{14} Worst



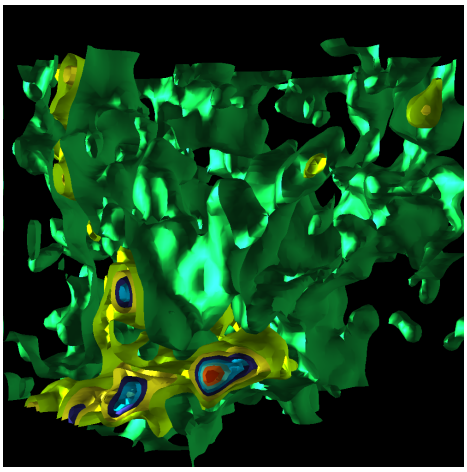
(y) VQ_{15} Best



(z) VQ_{15} Worst

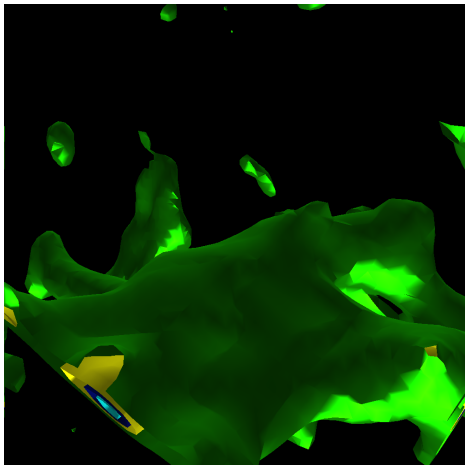


(aa) VQ_{16} Best

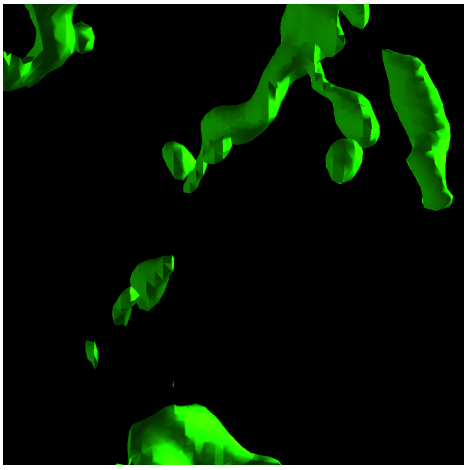


(ab) VQ_{16} Worst

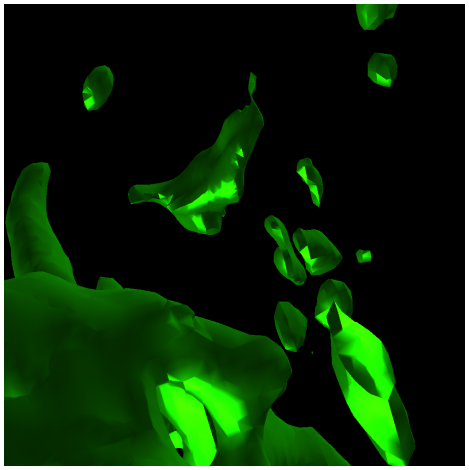
Figure A.47. The best and worst images for the implemented metrics on the ExaSky #2 timestep with 143,059 triangles.



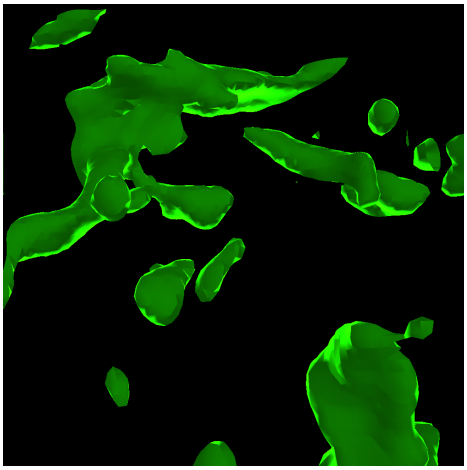
(a) VQ_1 Best



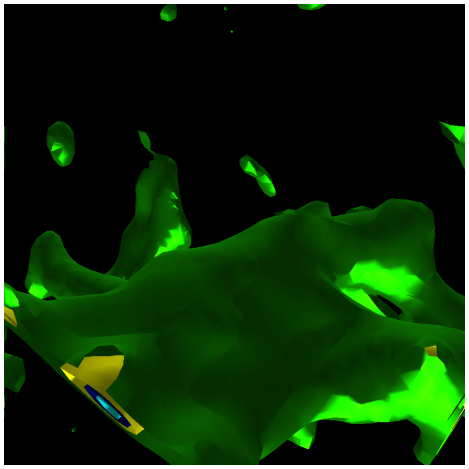
(b) VQ_1 Worst



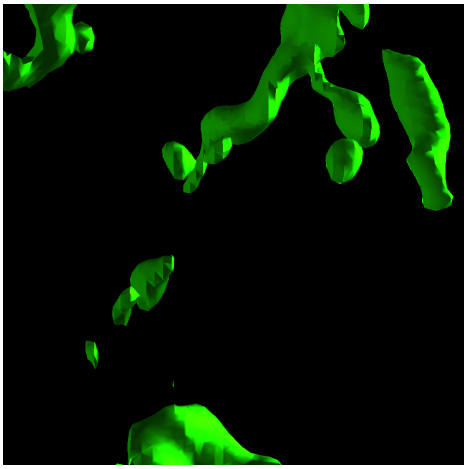
(c) VQ_2 Best



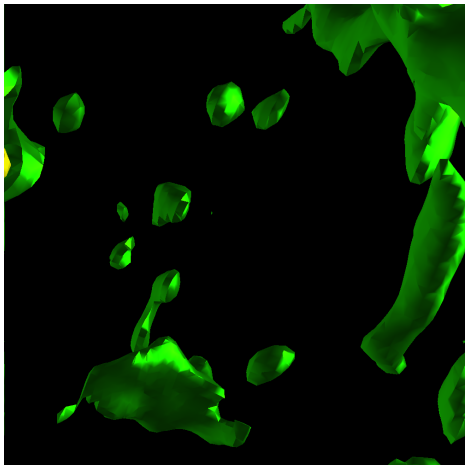
(d) VQ_2 Worst



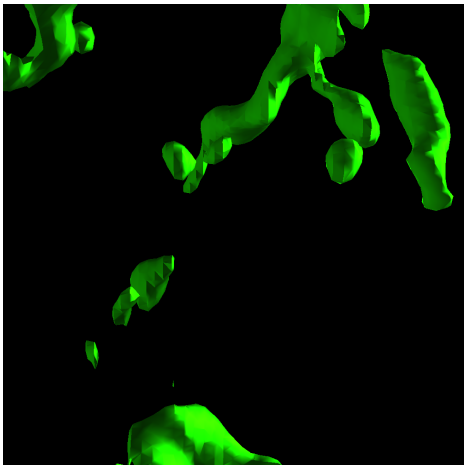
(e) VQ_3 Best



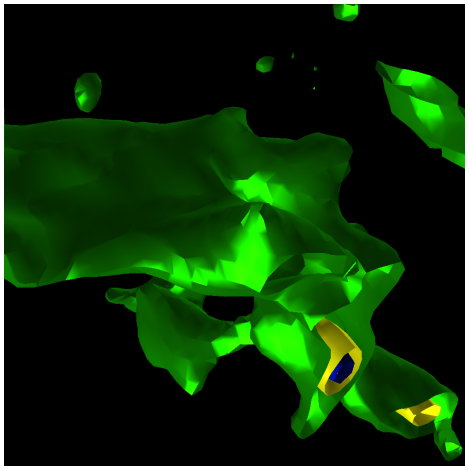
(f) VQ_3 Worst



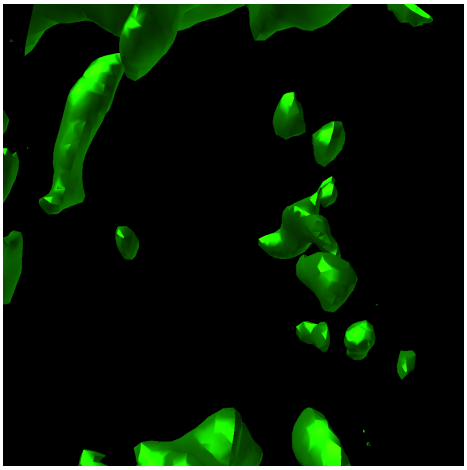
(g) VQ_4 Best



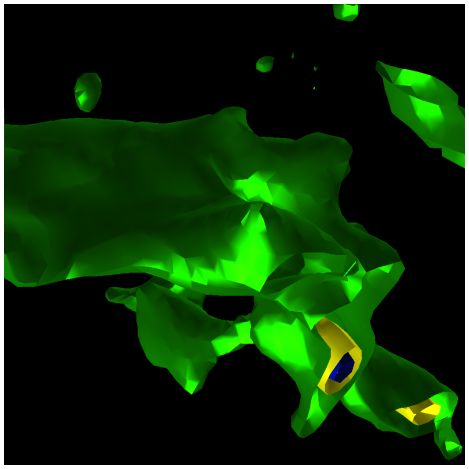
(h) VQ_4 Worst



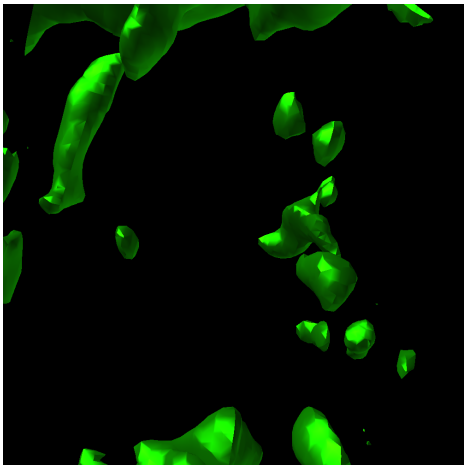
(i) VQ_5 Best



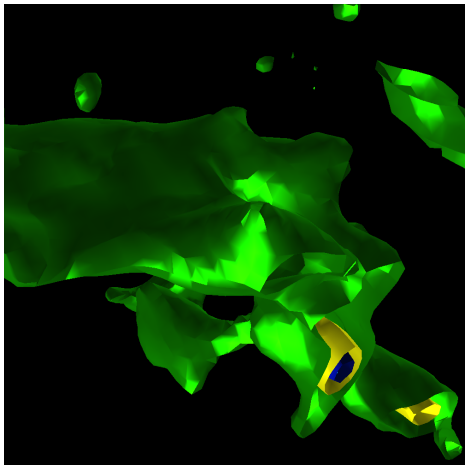
(j) VQ_5 Worst



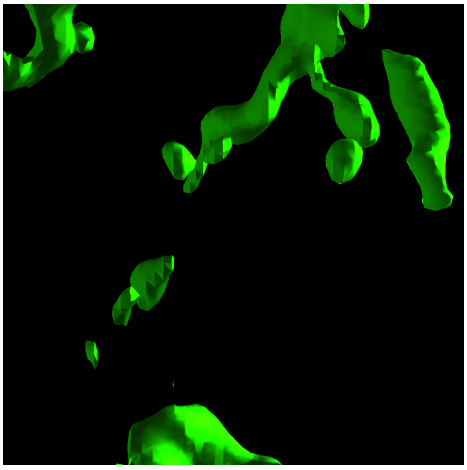
(k) VQ_6 Best



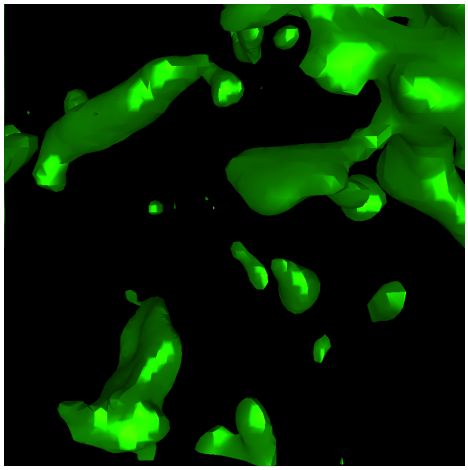
(l) VQ_6 Worst



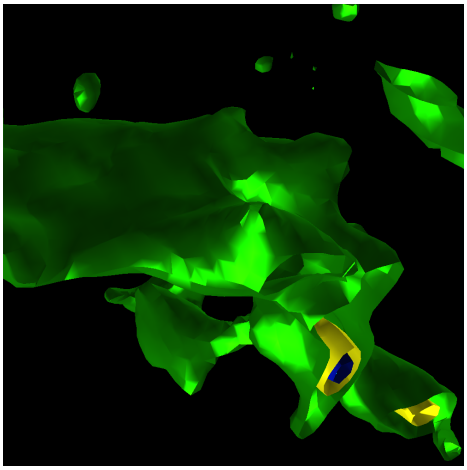
(m) VQ_7 Best



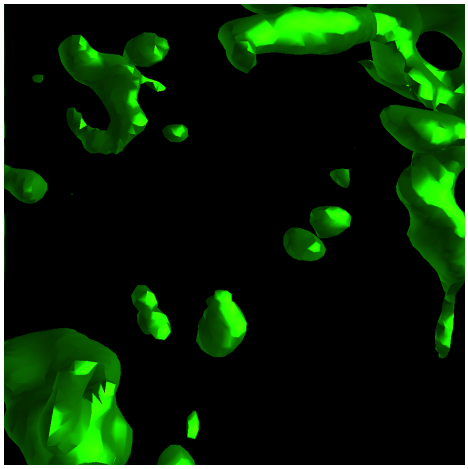
(n) VQ_7 Worst



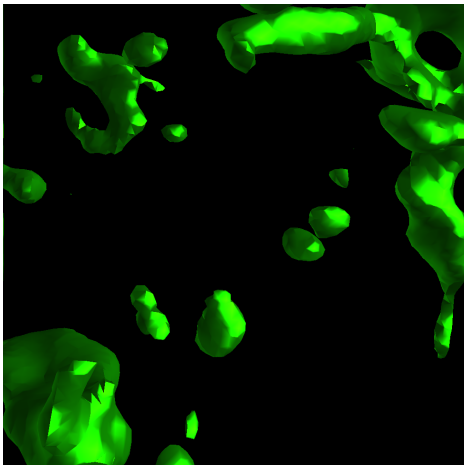
(o) VQ_{10} Best



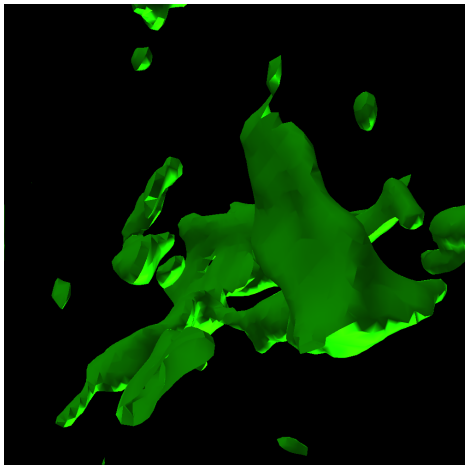
(p) VQ_{10} Worst



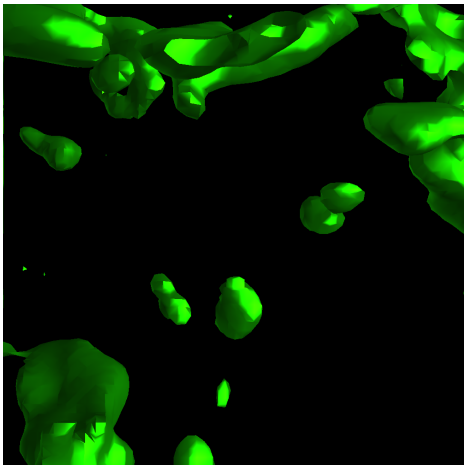
(q) VQ_{11} Best



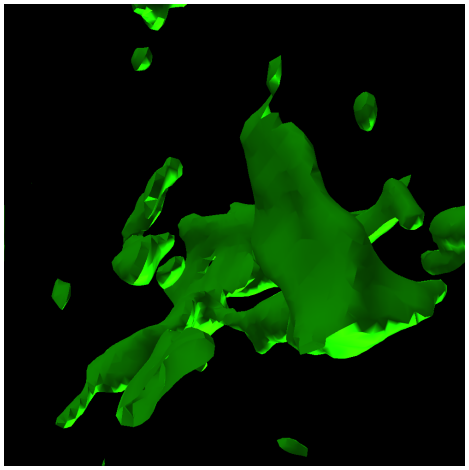
(r) VQ_{11} Worst



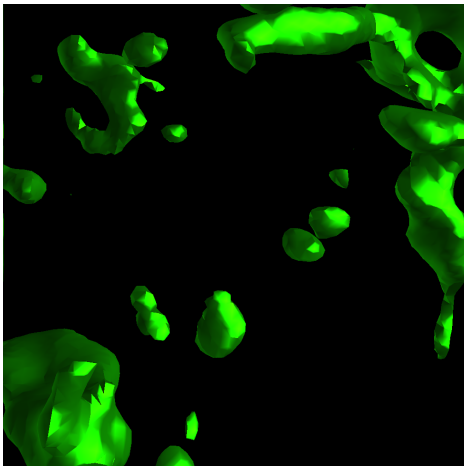
(s) VQ_{12} Best



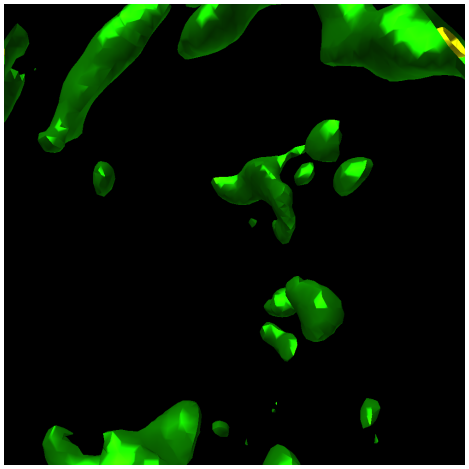
(t) VQ_{12} Worst



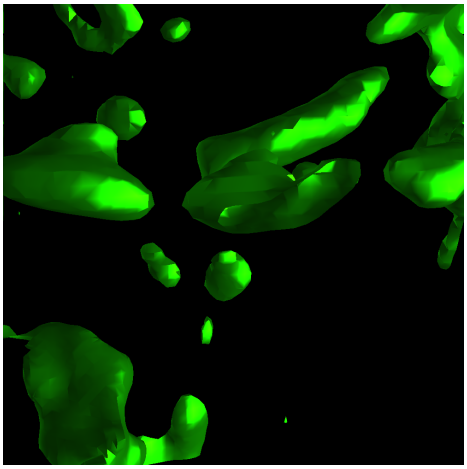
(u) VQ_{13} Best



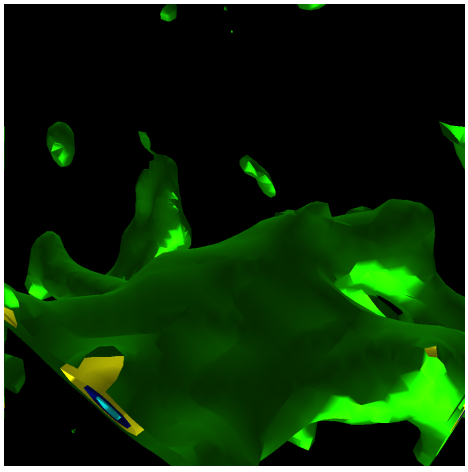
(v) VQ_{13} Worst



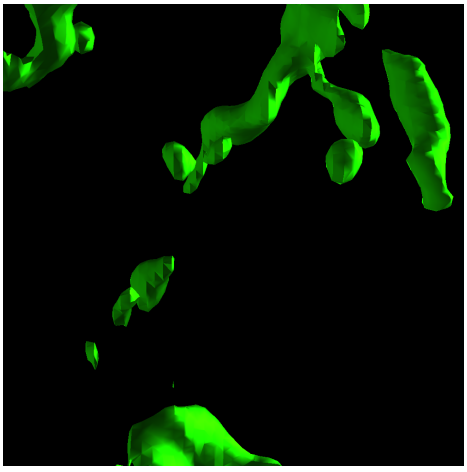
(w) VQ_{14} Best



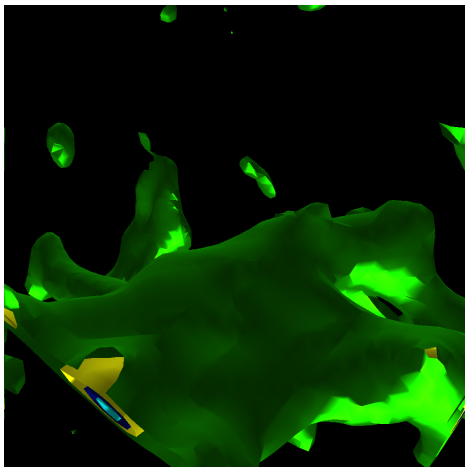
(x) VQ_{14} Worst



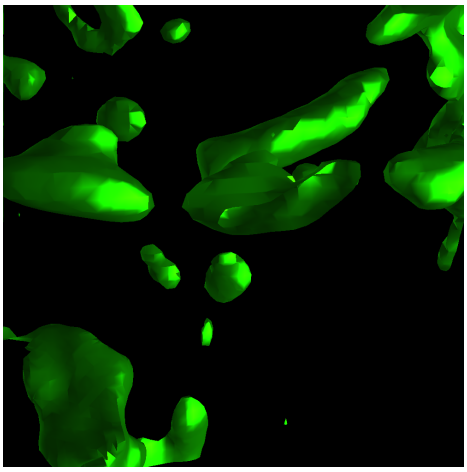
(y) VQ_{15} Best



(z) VQ_{15} Worst

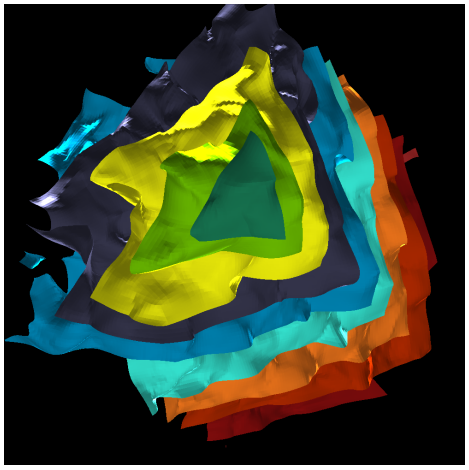


(aa) VQ_{16} Best

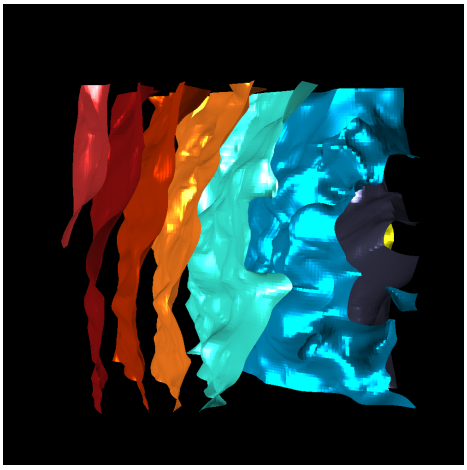


(ab) VQ_{16} Worst

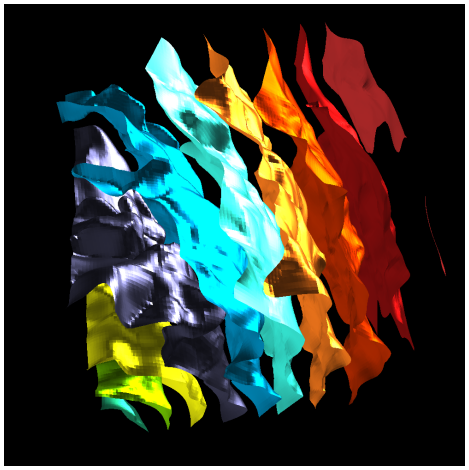
Figure A.48. The best and worst images for the implemented metrics on the ExaSky #3 timestep with 19,280 triangles.



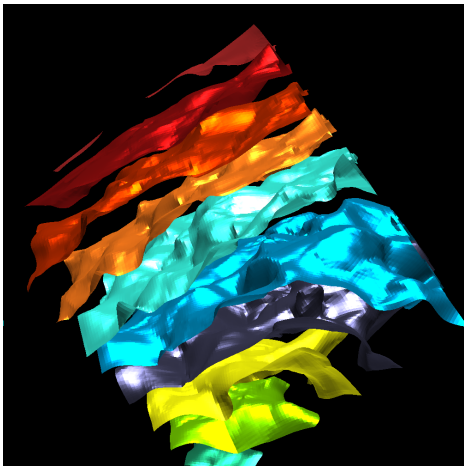
(a) VQ_1 Best



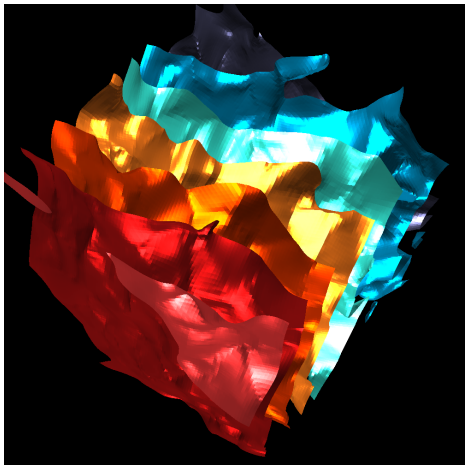
(b) VQ_1 Worst



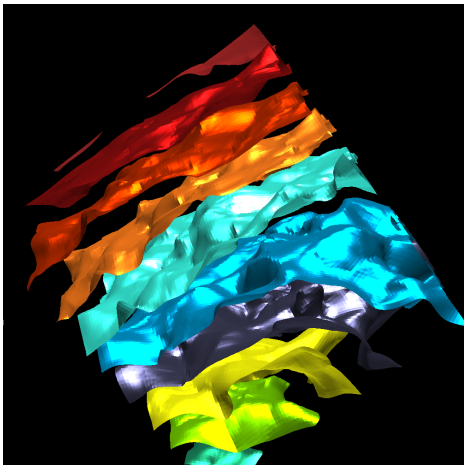
(c) VQ_2 Best



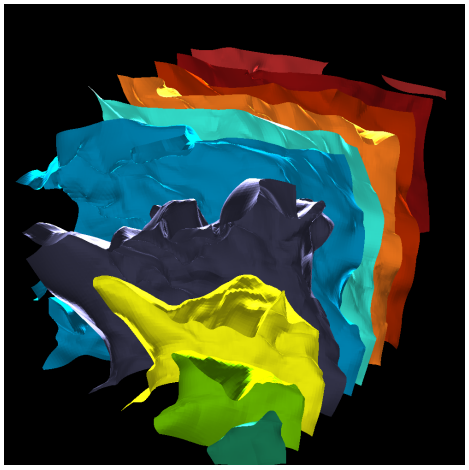
(d) VQ_2 Worst



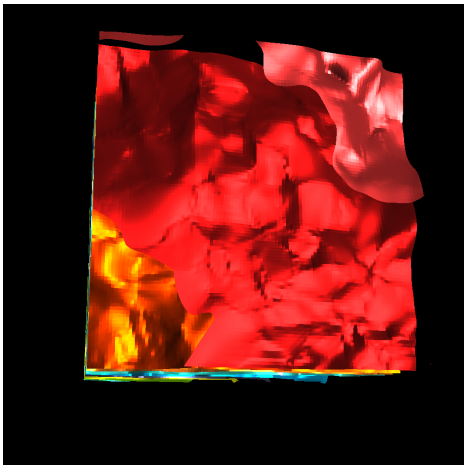
(e) VQ_3 Best



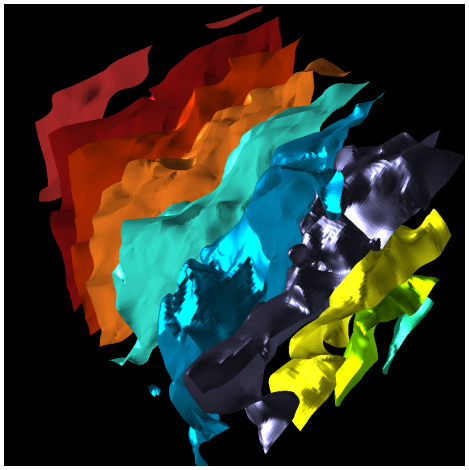
(f) VQ_3 Worst



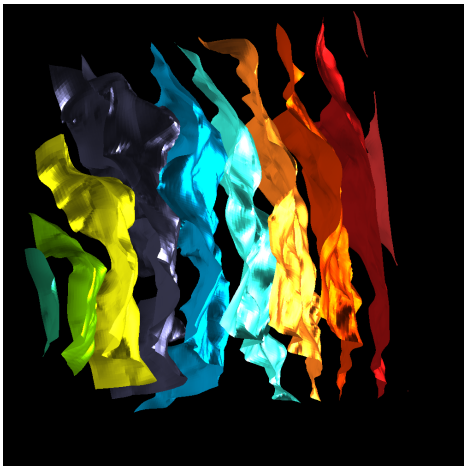
(g) VQ_4 Best



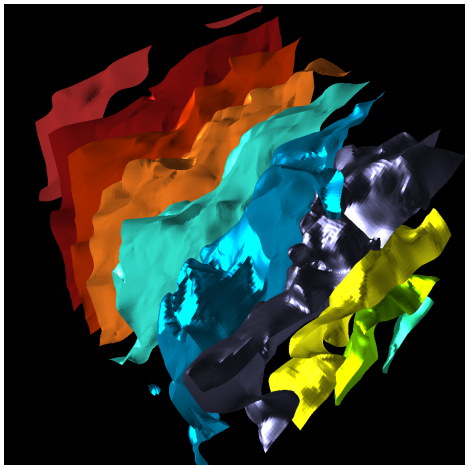
(h) VQ_4 Worst



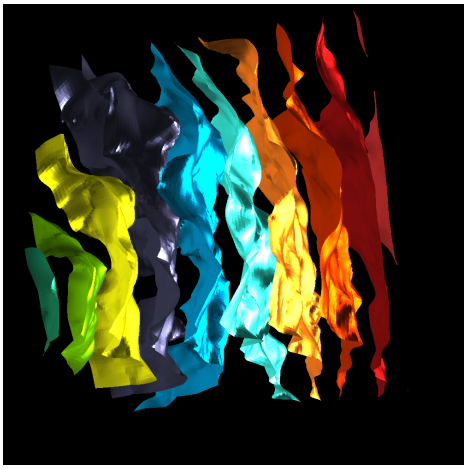
(i) VQ_5 Best



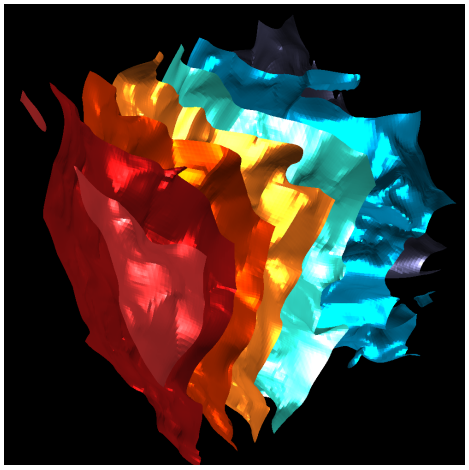
(j) VQ_5 Worst



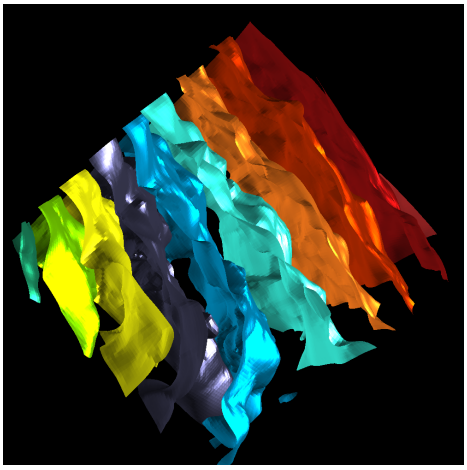
(k) VQ_6 Best



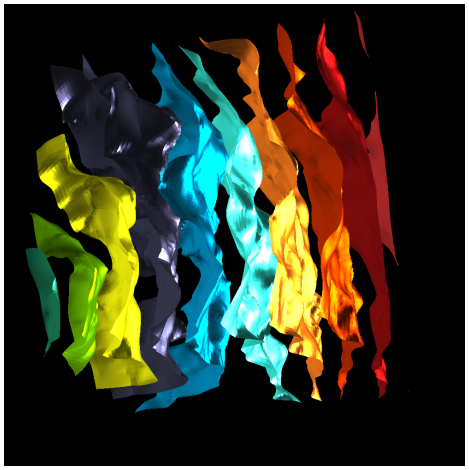
(l) VQ_6 Worst



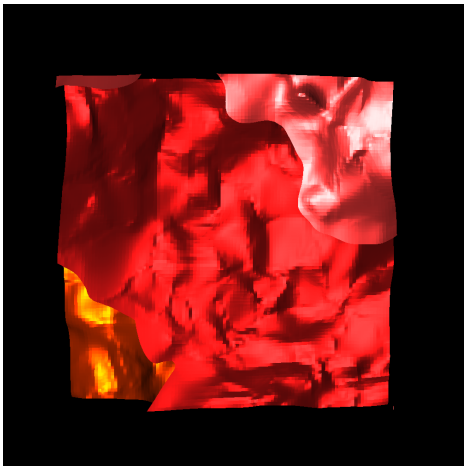
(m) VQ_7 Best



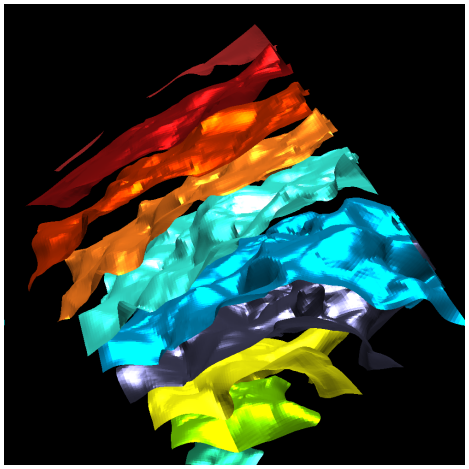
(n) VQ_7 Worst



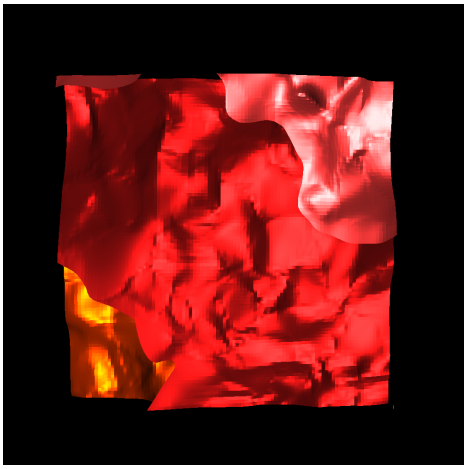
(o) VQ_{10} Best



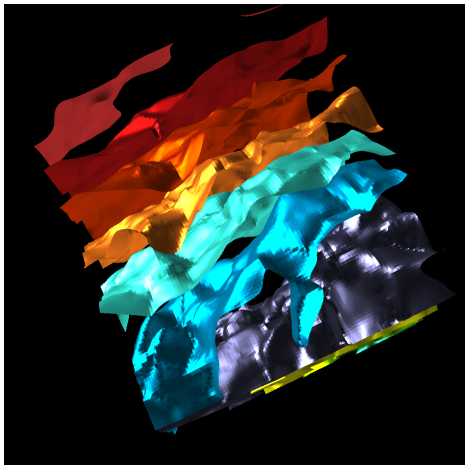
(p) VQ_{10} Worst



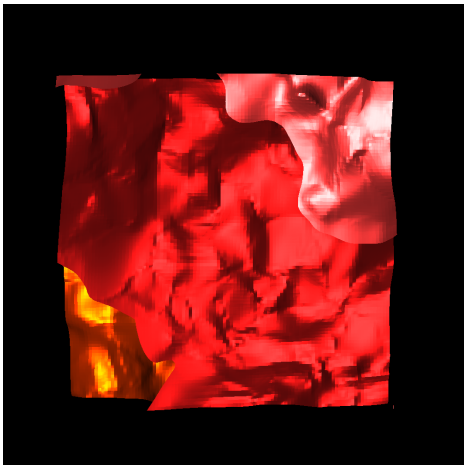
(q) VQ_{11} Best



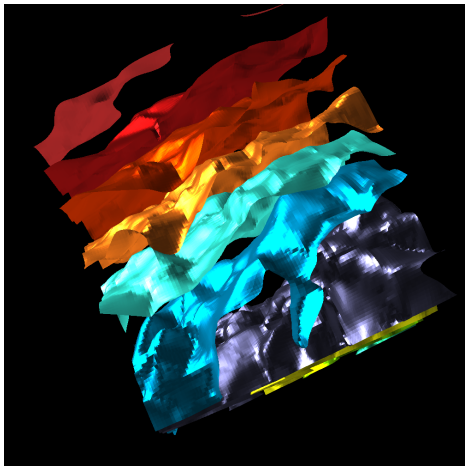
(r) VQ_{11} Worst



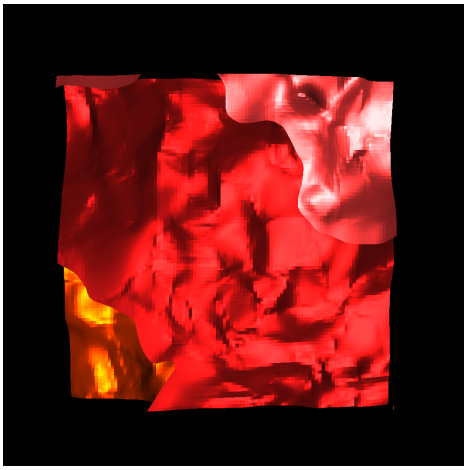
(s) VQ_{12} Best



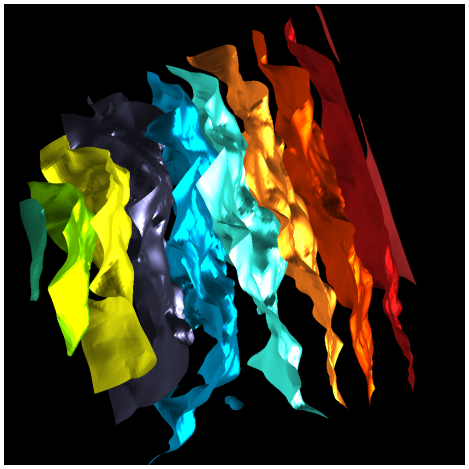
(t) VQ_{12} Worst



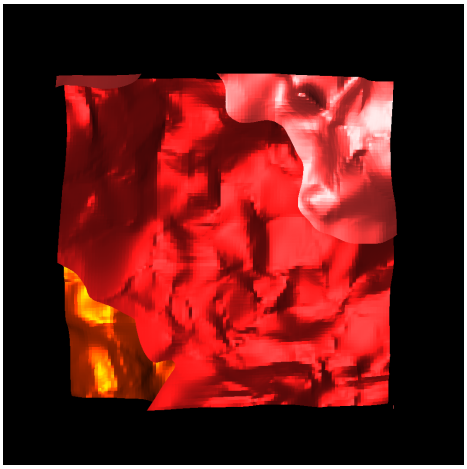
(u) VQ_{13} Best



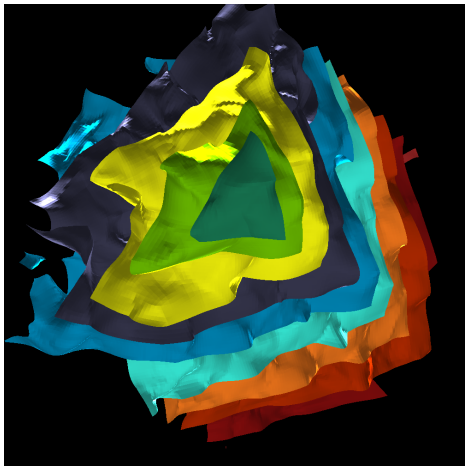
(v) VQ_{13} Worst



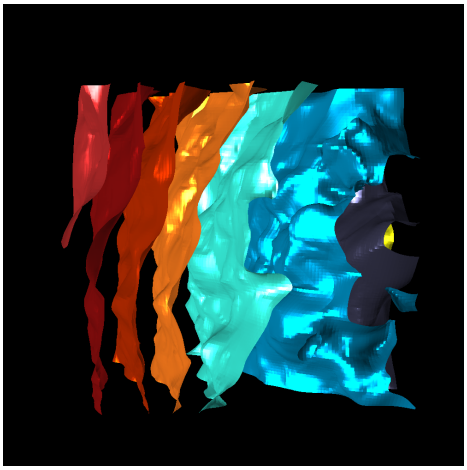
(w) VQ_{14} Best



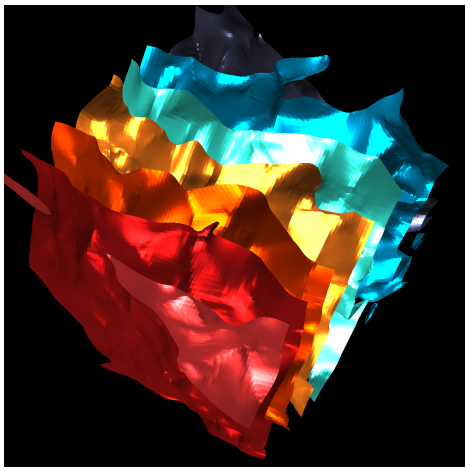
(x) VQ_{14} Worst



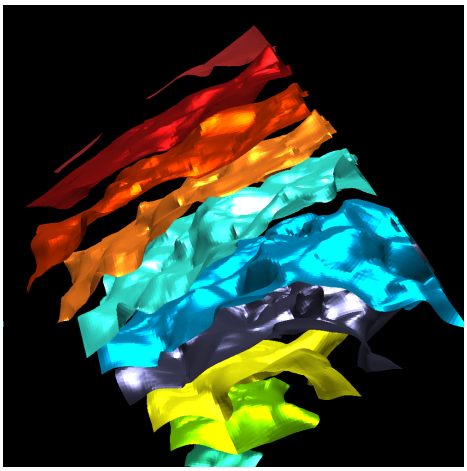
(y) VQ_{15} Best



(z) VQ_{15} Worst

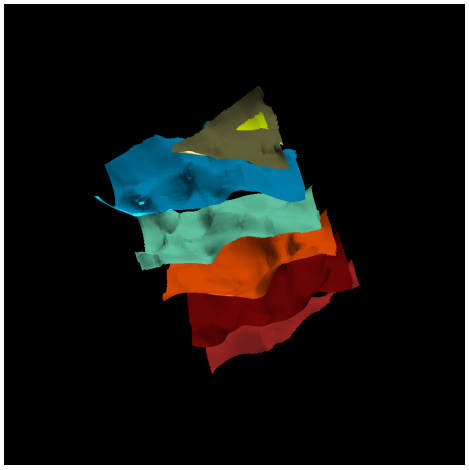


(aa) VQ_{16} Best

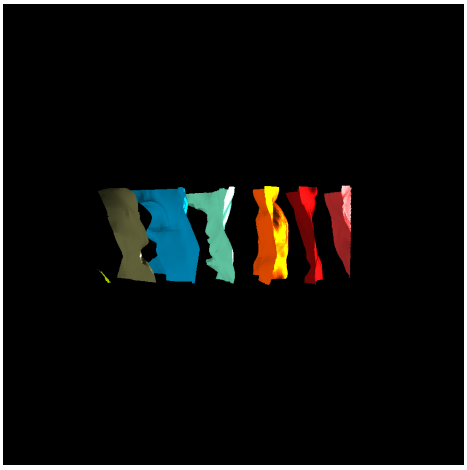


(ab) VQ_{16} Worst

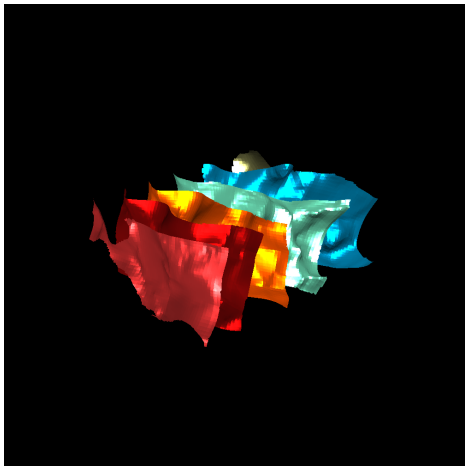
Figure A.49. The best and worst images for the implemented metrics on the ExaConstit #1 timestep with 938,862 triangles.



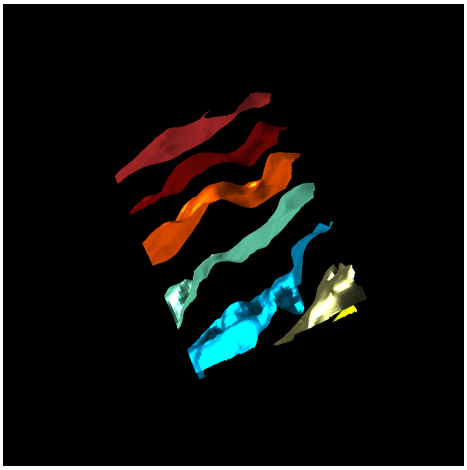
(a) VQ_1 Best



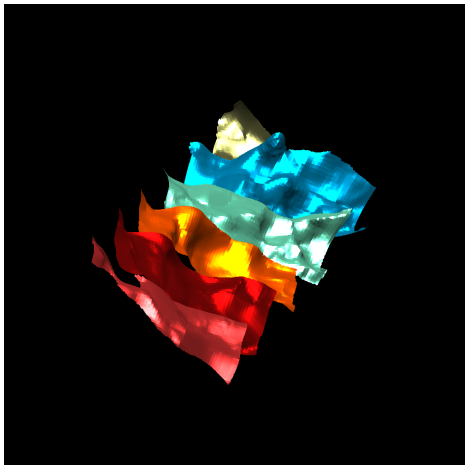
(b) VQ_1 Worst



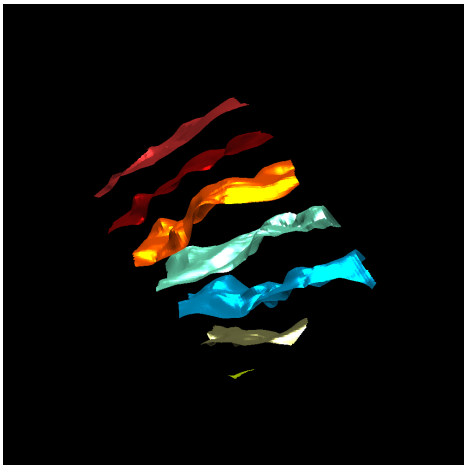
(c) VQ_2 Best



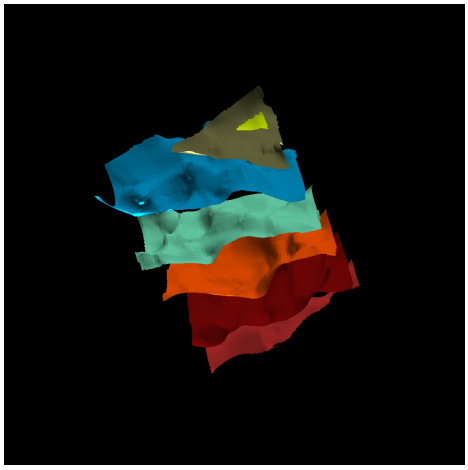
(d) VQ_2 Worst



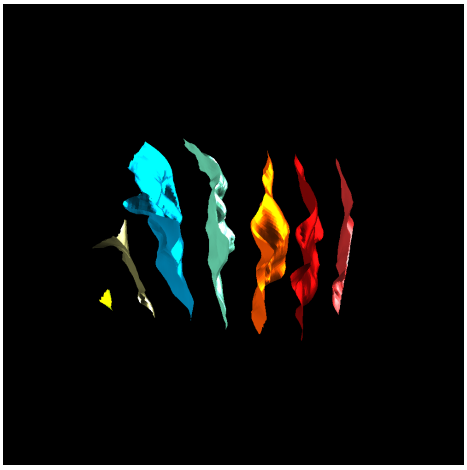
(e) VQ_3 Best



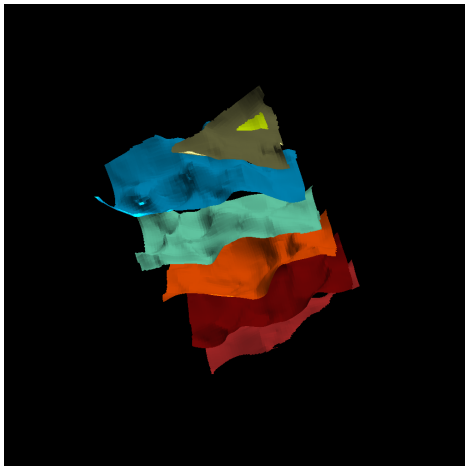
(f) VQ_3 Worst



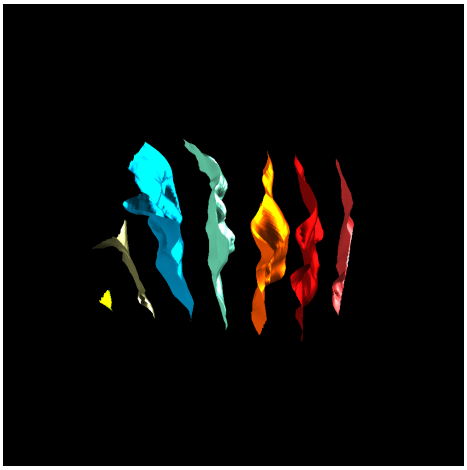
(g) VQ_4 Best



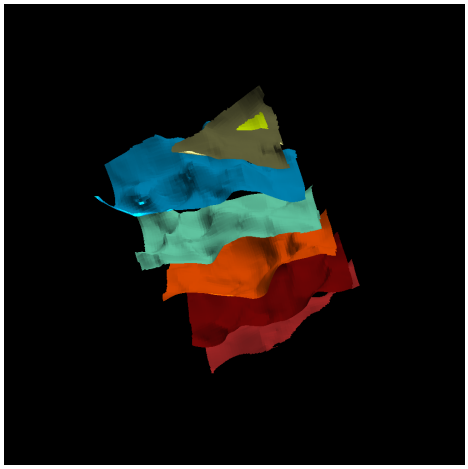
(h) VQ_4 Worst



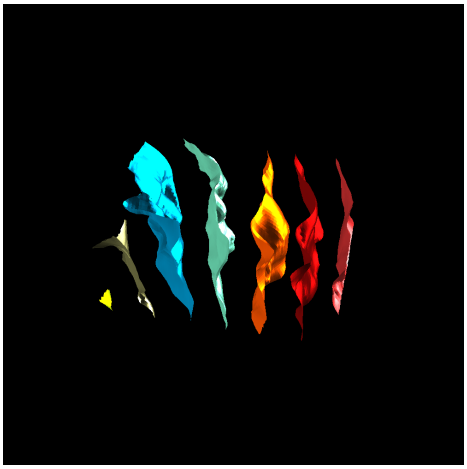
(i) VQ_5 Best



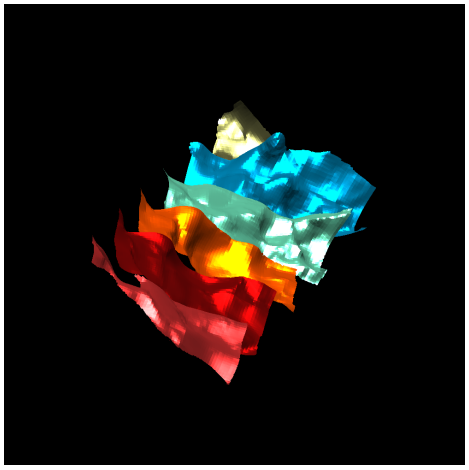
(j) VQ_5 Worst



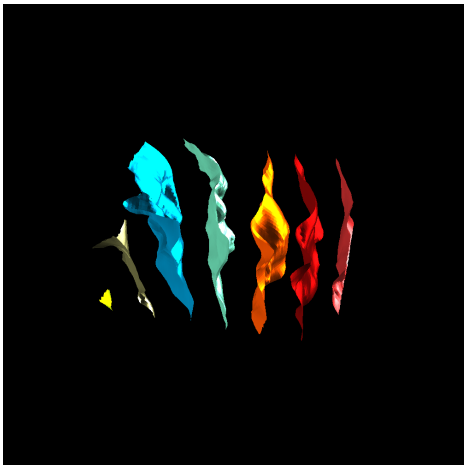
(k) VQ_6 Best



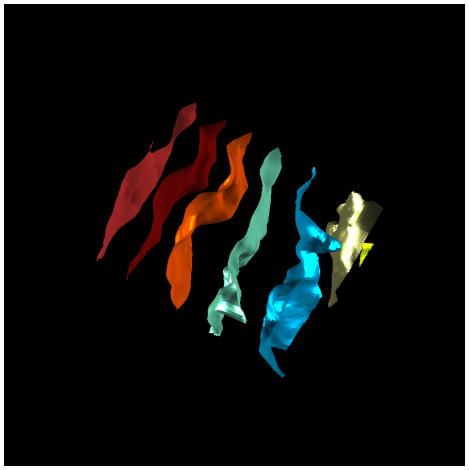
(l) VQ_6 Worst



(m) VQ_7 Best

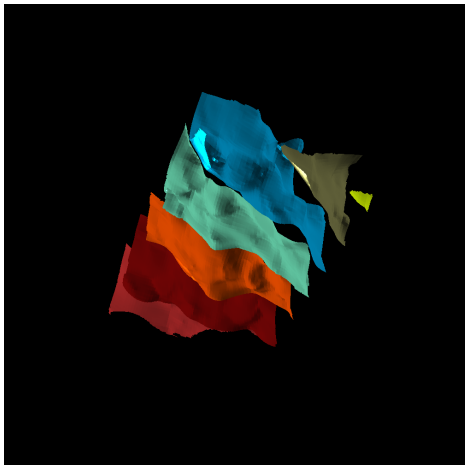


(n) VQ_7 Worst



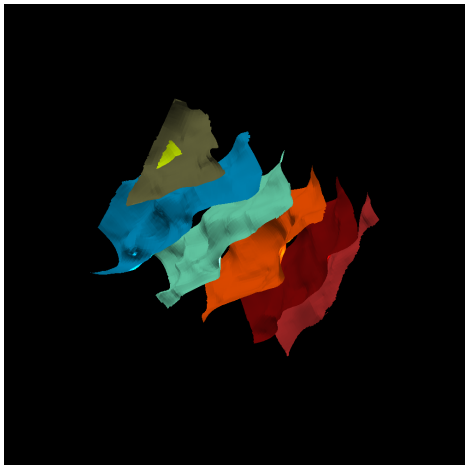
(o) VQ_{10} Best

(p) VQ_{10} Worst



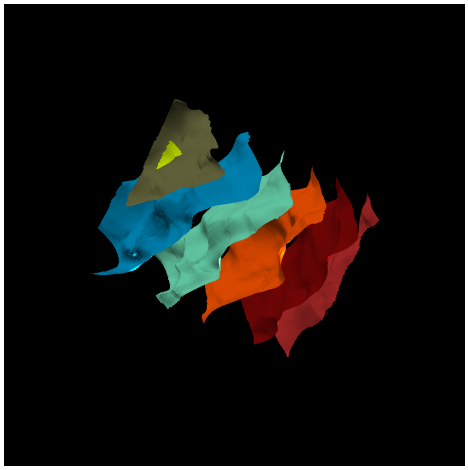
(q) VQ_{11} Best

(r) VQ_{11} Worst



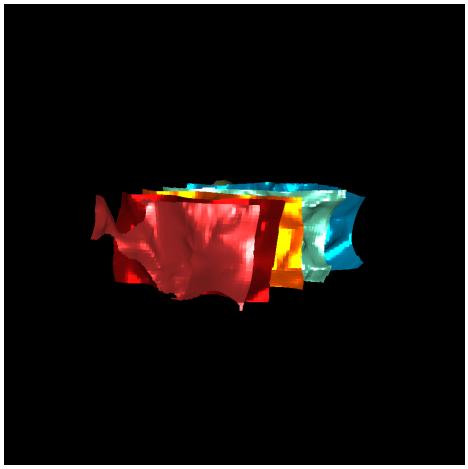
(s) VQ_{12} Best

(t) VQ_{12} Worst

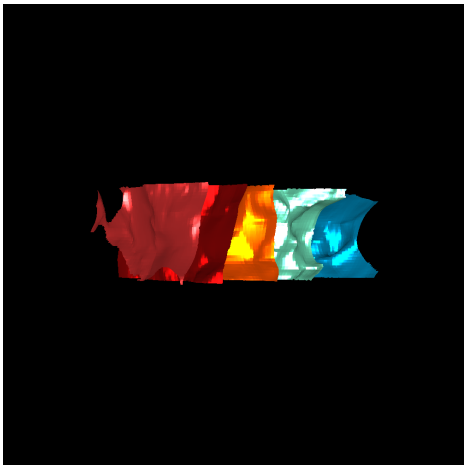


(u) VQ_{13} Best

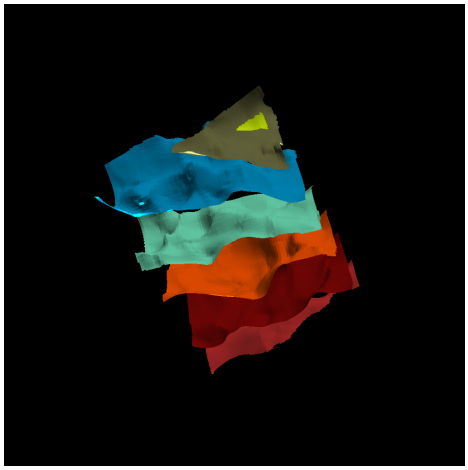
(v) VQ_{13} Worst



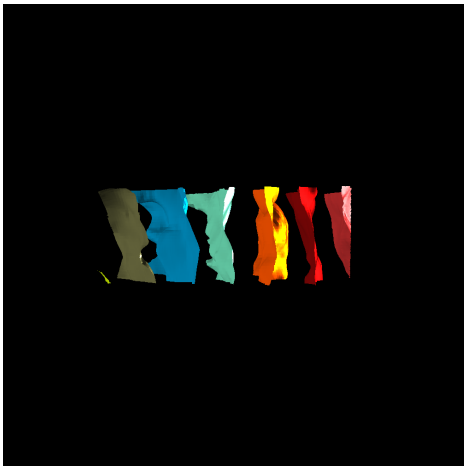
(w) VQ_{14} Best



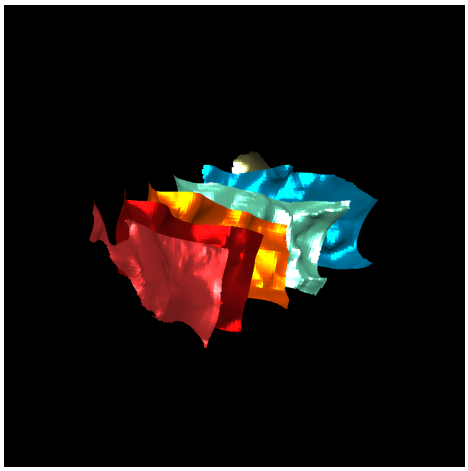
(x) VQ_{14} Worst



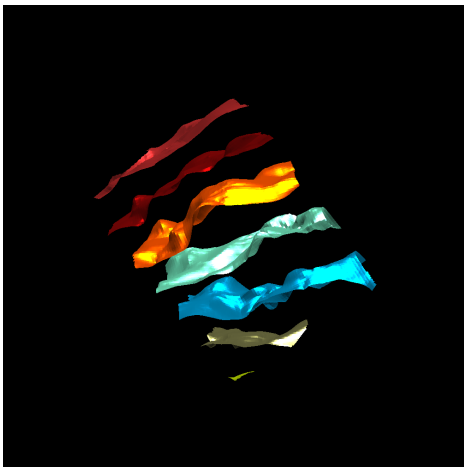
(y) VQ_{15} Best



(z) VQ_{15} Worst



(aa) VQ_{16} Best



(ab) VQ_{16} Worst

Figure A.50. The best and worst images for the implemented metrics on the ExaConstit #2 timestep with 135,109 triangles.

REFERENCES CITED

[1] Hurricane isabel simulation data. *http://vis.computer.org/vis2004contest/data.html* (2004), online.

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

[3] Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D. H., and 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.

[4] Akoglu, L., Tong, H., and Koutra, D. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery 29*, 3 (2015), 626–688.

[5] Almgren, A. S., Bell, J. B., Lijewski, M. J., Lukiä, Z., and Van Andel, E. Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal 765*, 1 (Feb 2013), 39.

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

[7] Arnheim, R. The power of center. University of California Press.

[8] Atanasov, A., Bungartz, H.-J., Frisch, J., Mehl, M., Mundani, R.-P., Rank, E., and van Treeck, C. Computational steering of complex flow simulations. *High Performance Computing in Science and Engineering, Garching/Munich 2009* (2010).

[9] Ayachit, U., Bauer, A., Geveci, B., O'Leary, P., Moreland, K., Fabian, N., and 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.

[10] Ayachit, U., Whitlock, B., Wolf, M., Loring, B., Geveci, B., Lonie, D., and 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.

[11] Bai, Z., Yang, R., Zhou, Z., Tao, Y., and Lin, H. Topology aware view path design for time-varying volume data. *Journal of Visualization 19*, 4 (2016), 797–809.

[12] Bajaj, C. L., Pascucci, V., and 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.

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

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

[15] Beazley, D., and Lomdahl, P. Lightweight computational steering of very large scale molecular dynamics simulations. In *Supercomputing '96:Proceedings of the 1996 ACM/IEEE Conference on Supercomputing* (1996), pp. 50–50.

[16] Belak, J., Turner, J., and Team, E. T. Exaam: Additive manufacturing process modeling at the fidelity of the microstructure. In *APS March Meeting Abstracts* (2019), vol. 2019, pp. C22–010.

[17] 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., and 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.

[18] Bennett, J. C., Bhagatwala, A., Chen, J. H., Seshadhri, C., Pinar, A., and Salloum, M. Trigger detection for adaptive scientific workflows using percentile sampling. *CoRR abs/1506.08258* (2015).

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

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

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

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

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

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

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

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

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

[28] Bordoloi, U., and 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.

[29] Bujack, R., Turton, T. L., Samsel, F., Ware, C., Rogers, D. H., and 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.

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

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

[32] Carson, R. A., Wopschall, S. R., Bramwell, J. A., et al. Exaconstit. Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2019.

[33] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. 41*, 3 (jul 2009).

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

[35] 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., and 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.

[36] Childs, H., Loring, B., Rogers, D., Larsen, M., Harrison, C., Rizzi, S., Whitlock, B., and 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.

[37] Childs, H., Ma, K.-L., Yu, H., Whitlock, B., Meredith, J., Favre, J., Klasky, S., Podhorszki, N., Schwan, K., Wolf, M., Parashar, M., and Zhang, F. In Situ Processing. In *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*. CRC Press/Francis–Taylor Group, Oct. 2012, pp. 171–198.

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

[39] Cook, A. W., Cabot, W. H., Williams, P. L., Miller, B. J., de Supinski, B. R., Yates, R. K., and Welcome, M. L. Tera-scalable algorithms for variable-density elliptic hydrodynamics with spectral accuracy. In *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing* (2005), IEEE, pp. 60–60.

[40] Coulaud, O., Dussere, M., and Esnard, A. Toward a distributed computational steering environment based on corba. In *Parallel Computing*, G. Joubert, W. Nagel, F. Peters, and W. Walter, Eds., vol. 13 of *Advances in Parallel Computing*. North-Holland, 2004, pp. 151–158.

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

[42] Davison de St. Germain, J., McCorquodale, J., Parker, S., and Johnson, C. Uintah: a massively parallel problem solving environment. In *Proceedings the Ninth International Symposium on High-Performance Distributed Computing* (2000), pp. 33–41.

[43] Dayal, J., Bratcher, D., Eisenhauer, G., Schwan, K., Wolf, M., Zhang, X., Abbasi, H., Klasky, S., and 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.

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

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

[46] Dorier, M., Sisneros, R., Peterka, T., Antoniu, G., and Semeraro, D. Damaris/viz: a nonintrusive, adaptable and user-friendly in situ visualization framework. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)* (2013), IEEE, pp. 67–75.

[47] DSS Los Alamos National Lab, . Cinema tutorial. In *International Conference for High Performance Computing, Networking, Storage and Analysis* (2019), SC '19.

[48] Dutagaci, H., Cheung, C. P., and Godil, A. A benchmark for best view selection of 3d objects. In *Proceedings of the ACM workshop on 3D object retrieval* (2010), pp. 45–50.

[49] Eisenhauer, G., Gu, W., Kindler, T., Schwan, K., Silva, D., and Vetter, J. Opportunities and tools for highly interactive distributed and parallel computing. Tech. rep., Proceedings of the Workshop, 1996.

[50] Eisenhauer, G., Schwan, K., Gu, W., and Mallavarupu, N. Falcon-toward interactive parallel programs: the on-line steering of a molecular dynamics application. In *Proceedings of 3rd IEEE International Symposium on High Performance Distributed Computing* (1994), pp. 26–33.

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

[52] Esnard, A., Richart, N., and Coulaud, O. A steering environment for online parallel visualization of legacy parallel simulations. In *2006 Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications* (2006), pp. 7–14.

[53] Fabian, N., Moreland, K., Thompson, D., Bauer, A. C., Marion, P., Gevecik, B., Rasquin, M., and Jansen, K. E. The paraview coprocessing library: A scalable, general purpose in situ visualization library. In *2011 IEEE Symposium on Large Data Analysis and Visualization* (2011), IEEE, pp. 89–96.

[54] Feixas, M., Sbert, M., and 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.

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

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

[57] Geist, G., Kohl, J. A., and Papadopoulos, P. M. Cumulvs: Providing fault toler. ance, visualization, and steer ing of parallel applications. *The International Journal of Supercomputer Applications and High Performance Computing 11*, 3 (1997), 224–235.

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

[59] Gumhold, S. Maximum entropy light source placement. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (New York, NY, USA, 2002), SIGGRAPH '02, Association for Computing Machinery, p. 215.

[60] Hackstadt, S., Harrop, C., and Malony, A. A framework for interacting with distributed programs and data. In *Proceedings. The Seventh International Symposium on High Performance Distributed Computing (Cat. No.98TB100244)* (1998), pp. 206–214.

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

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

[63] Itti, L., Koch, C., and 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.

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

[65] Jenks, G. Optimal data classification for choropleth maps occasional paper no 2. *University of Kansas, Department of Geography* (1977).

[66] Ji, G., and Shen, H.-w. Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1109–1116.

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

[68] Kamada, T., and 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.

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

[70] Kuhnert, J. Meshfree numerical schemes for time dependent problems in fluid and continuum mechanics. *Advances in PDE modeling and computation* (2014), 119–136.

[71] Larsen, M., Ahrens, J., Ayachit, U., Brugger, E., Childs, H., Geveci, B., and 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.

[72] Larsen, M., Brugger, E., Childs, H., Eliot, J., Griffin, K., and 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.

[73] Larsen, M., Harrison, C., Kress, J., Pugmire, D., Meredith, J. S., and 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.

[74] Larsen, M., Woods, A., Marsaglia, N., Biswas, A., Dutta, S., Harrison, C., and Childs, H. A flexible system for in situ triggers. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (New York, NY, USA, 2018), ISAV '18, Association for Computing Machinery, pp. 1–6.

[75] Lee, C. H., Varshney, A., and Jacobs, D. W. Mesh saliency. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 659–666.

[76] Lee, T.-Y., Mishchenko, O., Shen, H.-W., and Crawfis, R. View point evaluation and streamline filtering for flow visualization. pp. 83–90.

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

[78] Lessley, B., Li, S., and Childs, H. HashFight: A Platform-Portable Hash Table for Multi-Core and Many-Core Architectures. In *IS&T International Symposium on Electronic Imaging: Visualization and Data Analysis (VDA)* (Burlingame, CA, Jan. 2020), pp. 376–1–376–12.

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

[80] Li, S., Larsen, M., Clyne, J., and 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.

[81] Li, S., Marsaglia, N., Chen, V., Sewell, C. M., Clyne, J. P., and Childs, H. Achieving portable performance for wavelet compression using data parallel primitives. In *EGPGV* (2017), pp. 73–81.

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

[83] Ling, J., Kegelmeyer, W., Aditya, K., Kolla, H., Reed, K., Shead, T., and Davis, W. Using feature importance metrics to detect events of interest in scientific computing applications. pp. 55–63.

[84] 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., and 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.

[85] Liu, Y., Chen, G., Sun, M., Liu, S., and Tian, F. A parallel sla-based algorithm for global mesoscale eddy identification. *Journal of Atmospheric and Oceanic Technology 33*, 12 (2016), 2743 – 2754.

[86] Lorendeau, B., Fournier, Y., and 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.

[87] Ma, J., Tao, J., Wang, C., Li, C., Shene, C.-K., and Kim, S. H. Moving with the flow: an automatic tour of unsteady flow fields. *Journal of Visualization 22*, 6 (2019), 1125–1144.

[88] Ma, J., Walker, J., Wang, C., Kuhl, S., and Shene, C. K. Flowtour: An automatic guide for exploring internal flow features. In *2014 IEEE Pacific Visualization Symposium* (2014), pp. 25–32.

[89] Malakar, P., Vishwanath, V., Munson, T., Knight, C., Hereld, M., Leyffer, S., and 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.

[90] Marchesin, S., Chen, C., Ho, C., and Ma, K. View-dependent streamlines for 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov 2010), 1578–1586.

[91] 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., and 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.

[92] Marsaglia, N., Kawakami, Y., Schwartz, S. D., Fields, S., and Childs, H. An Entropy-Based Approach for Identifying User-Preferred Camera Positions. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)* (2021).

[93] Messina, P. *The Exascale Computing Project*. Computing in Science and Engineering 19, 3, 2017.

[94] Moreland, K., Maynard, R., Pugmire, D., Yenpure, A., Vacanti, A., Larsen, M., and Childs, H. Minimizing Development Costs for Efficient Many-Core Visualization Using MCD^3. *Parallel Computing 108* (Dec. 2021), 102834.

[95] 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., and Geveci, B. Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications 36*, 3 (May 2016), 48–58.

[96] Page, D., Koschan, A., Sukumar, S. R., Abidi, B., and Abidi, M. Shape analysis algorithm based on information theory. vol. 1, pp. 229–232.

[97] Palmer, S. E., Rosch, E., and Chase, P. Canonical perspective and the perception of objects.

[98] Parker, S. G., and 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.

[99] Pascucci, V., Laney, D. E., Frank, R. J., Scorzelli, G., Linsen, L., Hamann, B., and 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.

[100] Patchett, J., and Gisler, G. Deep water impact ensemble data set.

[101] 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.

[102] Plemenos, D., and Benayada, M. Intelligent display in scene modelling. new techniques to automatically compute good views. In *GraphiCon'96* (Saint Petersburg (Russia), July 1996).

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

[104] Polonsky, O., Patané, G., Biasotti, S., Gotsman, C., and Spagnuolo, M. What's in an image? *The Visual Computer 21*, 8 (Sep 2005), 840–847.

[105] Poulin, P., and 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.

[106] Poulin, P., Ratib, K., and 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–.

[107] 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., and 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.

[108] Pugmire, D., Yenpure, A., Kim, M., Kress, J., Maynard, R., Childs, H., and Hentschel, B. Performance-Portable Particle Advection with VTK-m. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (Brno, Czech Republic, June 2018), pp. 45–55.

[109] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[110] Richart, N., Esnard, A., and Coulaud, O. Toward a computational steering environment for legacy coupled simulations. In *Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07)* (2007), IEEE, pp. 43–43.

[111] Ruiz, M., Bardera, A., Boada, I., Viola, I., Feixas, M., and Sbert, M. Automatic transfer functions based on informational divergence. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 1932–1941.

[112] Salehi, M., and Rashidi, L. A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]. *SIGKDD Explor. Newsl. 20*, 1 (may 2018), 13–23.

[113] Salloum, M., Bennett, J. C., Pinar, A., Bhagatwala, A., and Chen, J. H. Enabling adaptive scientific workflows via trigger detection. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (New York, NY, USA, 2015), ISAV2015, Association for Computing Machinery, pp. 41–45.

[114] Sander, F., and Krueger, F. *Gestaltpsychologie und Kunsttheorie: ein Beitrag zur Psychologie architektonischer Gestalten*. Beck, 1932.

[115] Sbert, M., Feixas, M., Rigau, J., Viola, I., and Chover, M. Applications of information theory to computer graphics. In *Eurographics* (2002).

[116] Sbert, M., Plemenos, D., Feixas, M., and 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.

[117] Schroeder, W., Martin, K. M., and Lorensen, W. E. *The Visualization Toolkit (4th Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[118] Secord, A., Lu, J., Finkelstein, A., Singh, M., and Nealen, A. Perceptual models of viewpoint preference. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 109:1–109:12.

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

[120] Shahnas, M. H., Peltier, W. R., Wu, Z., and Wentzcovitch, R. The high-pressure electronic spin transition in iron: Potential impacts upon mantle mixing. *Journal of Geophysical Research: Solid Earth 116*, B8 (2011).

[121] Slawinska, M., Clark, M., Wolf, M., Bode, T., Zou, H., Laguna, P., Logan, J., Kinsey, M., and 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.

[122] Sokolov, D., and 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.

[123] Stein, C., Bennett, D., Farrell, P., and Ruttan, A. A steering and visualization toolkit for distributed applications. pp. 451–457.

[124] Stoev, S. L., and 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.

[125] Sun, M., Tian, F., Liu, Y., and Chen, G. An improved automatic algorithm for global eddy tracking using satellite altimeter data. *Remote Sensing 9*, 3 (2017).

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

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

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

[129] Tao, J., Ma, J., Wang, C., and Shene, C.-K. A unified approach to streamline selection and viewpoint selection for 3d flow visualization. *IEEE Transactions on Visualization and Computer Graphics 19*, 3 (2013), 393–406.

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

[131] 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.

[132] Treichler, S., Bauer, M., Bhagatwala, A., Borghesi, G., Sankaran, R., Kolla, H., Mccormick, P., Slaughter, E., Lee, W., Aiken, A., and Chen, J. H. S3d-legion: An exascale software for direct numerical simulation of turbulent combustion with complex multicomponent chemistry.

[133] Ullrich, P. A., and Zarzycki, C. M. Tempestextremes: a framework for scale-insensitive pointwise feature tracking on unstructured grids. *Geoscientific Model Development 10*, 3 (2017), 1069–1090.

[134] Vázquez, P.-P., Feixas, M., Sbert, M., and 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.

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

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

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

[138] Vishwanath, V., Hereld, M., Papka, M. E., Hudson, R., Jordan, G. C., and Daley, C. J. A. In situ data analysis and i / o acceleration of flash astrophysics simulation on leadership-class system using glean. In *Proc. SciDAC, Journal of Physics: Conference Series* (2011).

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

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

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

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

[143] Whitlock, B., Favre, J. M., and 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.

[144] Wood, J., Brodlie, K., and Walton, J. gviz - visualization and steering for the grid.

[145] Yamamoto, K., and Kageyama, A. *In-Situ Visualization with Membrane Layer for Movie-Based Visualization.* 06 2019, pp. 588–594.

[146] Yamaoka, Y., Hayashi, K., Sakamoto, N., and Nonaka, J. In situ adaptive timestep control and visualization based on the spatio-temporal variations of the simulation results. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (New York, NY, USA, 2019), ISAV '19, Association for Computing Machinery, pp. 12–16.

[147] Yenpure, A., Childs, H., and Moreland, K. Efficient Point Merge Using Data Parallel Techniques. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (Porto, Portugal, June 2019), pp. 79–88.

[148] Yi, H., Rasquin, M., Fang, J., and 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.

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

[150] Zhao, M., Held, I. M., Lin, S.-J., and Vecchi, G. A. Simulations of global hurricane climatology, interannual variability, and response to global warming using a 50-km resolution gcm. *Journal of Climate 22*, 24 (2009), 6653 – 6678.