()

Automatic In Situ Camera Placement for Isosurfaces of Large-Scale Scientific Simulations

N. Marsaglia[1], M. Mathai[1], S. Fields[1], and H. Childs[1]

[1]University of Oregon, Eugene, OR, USA

Abstract

High-performance computing trends are requiring in situ processing increasingly often. This work considers automating camera placement for in situ visualization, specifically of isosurfaces, which is needed when there is no human in the loop and no a priori knowledge of where to place the camera. Our approach utilizes Viewpoint Quality (VQ) metrics, which quantify which camera positions provide the most insight. We have two primary contributions. First, we introduce an approach parallelizing the calculation of VQ metrics, which is necessary for usage in an in situ setting. Second, we introduce an algorithm for searching for a good camera position that balances between maximizing VQ metric score and minimizing execution time. We evaluate our contributions with an in situ performance study on a supercomputer. Our findings confirm that our approach is viable, and in particular that we can find good viewpoints with small execution time.

1. Introduction

In situ processing 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 can also 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 is done with no human in the loop. This change is important: where the HITL model enabled domain scientists to bring their expertise to direct the visualization and analysis process, their absence requires new approaches for performing 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 for in situ processing before the simulation begins. 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. We feel all three strategies are useful, and require attention from the in situ community. With this work, we focus on the third strategy, considering the task of automated camera placement.

Camera placement is a critical task for scientific visualization. In a typical post hoc setting, the camera placement process starts with some default camera position (e.g., zoomed out with the camera translated down the Z-axis and pointed at the center of the data set). Next, a domain scientist uses a mouse to modify the camera position, and the visualization program shows them the scene from the new viewpoint. This sequence is performed repeatedly until the domain scientist locates a position that increases their insight.

In the context of an automation strategy, in situ systems need algorithms that identify useful camera positions, ideally positions 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, previous research by the scientific visualization and graphics communities have provided some insight. This research has focused on Viewpoint Quality (VQ) metrics, i.e. metrics that quantify the camera placement of a data set. Previous research has primarily considered the post-hoc setting, with one of two possible goals: saving time for domain scientists (since they would not have to go through the process of finding good camera positions themselves) or discovering unexpected camera positions (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). Our work is the first to consider this approach in an in situ setting.

There are two major challenges to deploying VQ metrics for in situ camera placement. The first challenge is selecting VQ met-

rics 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. As 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 of viewpoints.

The two contributions of this work are:

- 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. While the parallelization is straightforward, we demonstrate feasibility and also that the approach is performant. This is, to the best of our knowledge, the first-ever approach for parallelizing calculation of these metrics.
- 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" between these tradeoffs.

In all, this work provides a pragmatic and useful approach for efficient in situ automation of camera placements.

2. Related Work

As discussed in the introduction, Viewpoint Quality (VQ) metrics are functions that quantify the camera placement of a given data set. There has been a wide breadth of research on developing VQ metrics, in particular because these techniques have many applications, including scientific visualization, object recognition, scene navigation, and mesh simplification. Most recently these metrics have been surveyed by Bonaventura et al. [B*18], with previous surveys by Polonsky et al. [P*05], Dutağaci et al. [D*10], and Secord et al. [S*11a].

The survey by Bonaventura et al. [B*18] organizes the metrics into five categories based on what aspect of the data the metrics incorporate in their calculations. These categories are: area [Ple91, PB96, V*01, SFR*02, B*11, S*05, F*09], depth [SS02, S*11a], silhouette [P*05, P*03, V*09, S*11a], image stability [BS05, BR82, Bla87, F*09, Lin91, V*01], and surface curvature [P*03, P*05, L*05, I*98, SP05, F*09]. The majority of the surveyed works above are based on use cases that do not involve scientific data.

There have been several works focusing on scientific data. Takahashi et al. [T*05a, T*05b] developed several VQ metrics for volume rendering, including metrics based on weight and unweighted isosurfaces and interval volumes. Ji and Shen [JS06] also consider viewpoint selection for volume rendering, producing a static approach that utilizes opacity entropy, color entropy, and surface curvature maximization. They then employ dynamic programming in conjunction with their static approach to determine viewpoint selection for time-varying volume data. Work from Tao et al. [T*09] used feature-based VQ metrics to determine the best viewpoint of volume renderings. Another work from Tao et al. [T*16] used a voting-based strategy that compares the input volume rendering with a database of visually similar images that they believe to be preferred viewpoints of visualization experts. Naraoka et al. [N*07] applied illumination entropy to volume renderings to determine the optimal light placement when the camera placement is fixed. For flow visualization, Lee et al. [L*11] calculated the entropy of vector fields to determine both seed and camera placement. In another approach, Tao et al. [T*13] utilize mutual information to define seed placement as well as optimal viewpoints and camera pathlines. Most recently, Marsaglia et al. [M*21] developed metrics that apply entropy to the visible field data, visible depth of the field data, and the visible shading coefficients of the visible data of isosurface images.

In terms of evaluation, Bonaventura et al. [B*18] tested the surveyed metrics using the Dutağaci benchmark [D*10], a benchmark composed of viewpoint preferences of 26 human subjects of 68 recognizable 3D objects. For scientific data, Marsaglia et al. [M*21] performed a user study of large data visualization experts, asking for a preferred image between a data set rendered from two different camera positions. They found that their entropy metrics align with user preference 68% of the time, and they were particularly effective at identifying disliked camera positions. This work is key to our own research, since it establishes that certain VQ metrics truly do predict user preference for large data visualization. Further, while our work considers a broad set of VQ metrics, this set includes the entropy metrics from Marsaglia et al. In other words, the Marsaglia user study establishes that the parallelization and search approaches we introduce in this work will produce useful results.

With respect to automating camera placement, previous works have primarily aimed to optimize volume rendering. Bordoloi and Shen [BS05] argued that the best viewpoints for volume rendered data must (1) display voxels with high noteworthiness factors and (2) viewpoints must contain an high amount of information. Correa and Ma [CM09] implemented the first criterion, allowing users to determine importance metrics that would then be used to create a transfer function to highlight the chosen intervals. That said, in the end, this work automated designing a transfer function from some viewpoint, rather than automating the viewpoint selection itself. Viola et al. [V*04, V*06] took a similar path, also utilizing importance metrics to create a transfer function, but then implemented the VQ metric Viewpoint Mutual Information (VMI) to automatically determine the best viewpoint. Finally, Yao and Wang [XY15] employed AI to determine both the transfer function and best camera placement.

In our study, we incorporate the same eleven VQ metrics Marsaglia et al. evaluated in their user preference study [M*21]. They selected these metrics based on their potential for in situ: small memory footprint, low communication overhead, and quick execution time. The eleven metrics are summarized in Table 1. Additionally, since each of these VQ metrics operate on surface data, all of the data sets used in this work are first transformed into isosurfaces before being rendered and evaluated.

3. Our Method

This section describes our method in two parts: Section 3.1 describes our parallelization for the calculation of VQ metrics, while Section 3.2 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,

Table 1: *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 [M*21].
DDS Entropy	The sum of the three VQ metric scores: Data Entropy + Depth Entropy + Shading Entropy [M*21]. 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 [M*21].
Max Depth	Maximizes the visible depth from some viewpoint [S*11a].
Plemenos and Benayada (PB)	Maximizes the number of visible triangles as well as the resolution of the rendered image [PB96].
Projected Area	Maximizes the visible projected area of the data set from some viewpoint [PB96].
Shading Entropy	Maximizes the entropy of the visible shading coefficients from some viewpoint [M*21].
Viewpoint Entropy	Maximizes the entropy of the projected area from some viewpoint [V*01].
Visibility Ratio	Maximizes the ratio of visible surface area over the total surface area.
Visible Triangles	Maximizes the total number of visible triangles from some viewpoint [PB96].
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 [S*05].

and it uses our parallelization approach to quickly calculate these metrics in a distributed-memory, in situ setting.

3.1. 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 isosurface data 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 identifiers instead of colors. For this metric, if a triangle with identifier 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 [L*17], which makes use of VTK-m [M*16]. 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 primitives, which then get executed efficiently on devices using an appropriate backend (CUDA, OpenMP, Kokkos). Our approach built on the parallel rendering infrastructure within Ascent/VTK-m, which follows the traditional approach: 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 with a pixel resolution of 1024×1024. For our approach, we adapted Ascent's rendering workflow to have the requisite information to carry out a VQ metric calculation (e.g., triangle identifiers), had Ascent render and composite the data from some viewpoint, and then analyzed the result using VQ metrics.

The VQ metric 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). VTK-m achieves performance and portability by writing visualization algorithms as VTK-m worklets that utilize parallel primitives. Parallel primitives can then be mapped to any architectures respective parallel routines, enabling efficient usage of the processor's architecture. We re-wrote each viewpoint quality metrics as VTK-m worklets in order to optimize their shared memory performance. Listing 1 shows C++ code that uses VTK-m to perform the local calculations for the VQ metric Max Depth.

Listing 1: *C++ code that uses VTK-m to perform the local calculations for the VQ metric Max Depth. As input, this templated function receives a rendering of the data set from some viewpoint, where, instead of RGB values, the pixels contain depth values, namely the distance from the camera to the visible geometry. The code for object MaxValueWithChecks can be found in Listing 2 at the end of the paper.*

```
template <typename T>
T calculateMaxDepth(const vtkm::cont::
    ArrayHandle<T> &depthData)
{
  T depth = -1.0 * (T)std::numeric_limits<
      int>::max();

  if (depthData.GetNumberOfValues() > 0)
  {
    MaxValueWithChecks<T> max{
      -1.0 * (T)std::numeric_limits<int>::
          max(),
      (T)std::numeric_limits<int>::max()};
    depth = vtkm::cont::Algorithm::Reduce(
        depthData, depth, max);
  }
  return depth;
}
```

As shown in Figure 1, the VQ metrics all follow one of two execution patterns. The execution patterns differ in the amount of global communications, as some VQ metrics require one such communication, while others require two. 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.

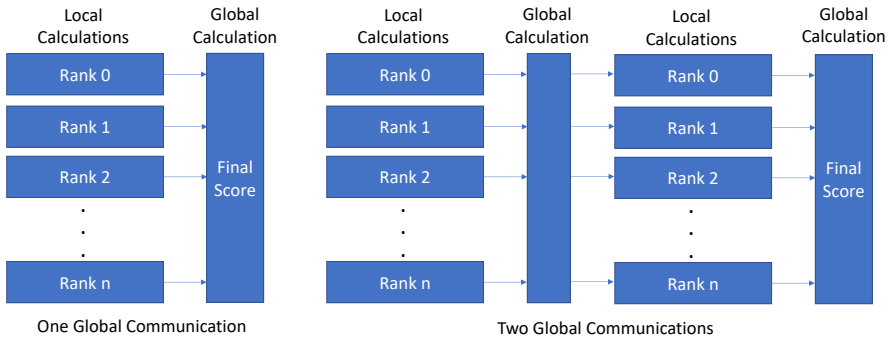

Figure 1: *The VQ metrics receive as input a rendering of the data set from some viewpoint, with the pixels in this rendering containing informative values (such as depth information, i.e. the distance from the camera to the visible geometry) instead of traditional RGB values. Using this input, the implemented metrics require either one or two global coordinations when computing a viewpoint score following any necessary local calculations, as shown.*

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.

3.2. 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 find good camera locations with few evaluations. Section 4.2.1 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 [A*14], 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, creating a search space of size $m * n$. A user can then decide a camera budget for how many camera placements to evaluate with each search algorithm. Camera placements are not the only way to formulate budgets, as we could have used total time as well. For our evaluation, a fixed number of placements made it easier to compare across data sets, but time may be a better choice for other settings.

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). The algorithms are:

- **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 1. This ensures that each additional camera position is placed into the largest unexplored region of possible camera positions.
- **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. This user-specified spacing is what leads to variants of the algorithm, with these variants having very different behavior, as sampling rate 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. To determine what sample rate to use, we tested all prime numbers that were less than or equal to half the diagonal of our sample space ($\frac{\sqrt{m^2+n^2}}{2}$), and selected representative samples rates.The sample rates selected for evaluation in Section 4.2.2 are 7, 23, and 43.
- **Random Search**: each new camera position is placed at a random location, as shown in Algorithm 3.
- **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 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 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 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 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. Results

Our results are organized into three phases. Phase 1 focuses on evaluating the performance of our parallelization approach for individual Viewpoint Quality metrics. 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. Please note that in this work we use "viewpoints" and "camera positions" interchangeably.

4.1. Phase 1: Parallel In Situ Performance of Individual Metrics

To evaluate parallel in situ performance, we ran the Lulesh proxy application that simulates a Sedov blast problem 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 ran for 100 cycles and we evaluated 10 camera positions per cycle. Each MPI task had a data size of 342^3, for a total data size of 1026^3.

For this phase of our experiment, Lulesh was executed for 100 cycles and evaluated 10 viewpoints per cycle with each VQ metric in situ. Table 2 contains the average execution time it took each VQ metric to evaluate a single viewpoint in situ. The major result from this phase of our experiment is that the rendering stage dominates the majority of the computation, and the execution of VQ metrics adds 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. The rest of this section informs holistic performance, with Phase 2 informing how many camera positions are necessary and Phase 3 considering overall in situ encumbrance. since it will consider overall in situ encumbrance.

4.2. Phase 2: Post Hoc Evaluation of 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.

4.2.1. 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 1. For data sets, we used the same ten data sets from the Marsaglia et al. [M*21] 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 [PG17].
- Constit: A material sciences data set that probes the deformation response of polycrystalline materials [C*19].
- ExaAm Truchas: A materials science data set that looks at effects within micro-structures of Additive Manufacturing (AM) [B*19].
- ExaSky Nyx: A cosmological data set that looks at gas dynamics [A*13].
- Fluid Dynamics: A fluid dynamics data set that models a cylindrical flow of water [Kuh14].
- Hurricane: A weather data set of Hurricane Isabel [K*04].
- Mantle: An earth sciences data set that models the Earth's mantle [S*11b].
- Miranda: A hydrodynamics data set of large-scale turbulence [C*05].
- S3D-N2: A combustion data set of field data N2 [T*17].
- S3D-UVEL: A combustion data set of field data U Velocity [T*17]

For each data set, we evaluated ten-thousand viewpoints using each VQ metric. The viewpoints were constructed in the same way as the search algorithms in Section 3.2: along the surface of a sphere, and taking even increments in ϕ and θ in Spherical Coordinates ($\phi_m = \theta_n = 100$). For each VQ metric, the ten-thousand scores were then rendered as heatmaps (high scores are white, low scores are dark orange and black):

- Figure 2 shows heatmaps for two data sets, including thumbnails of the images with high or low scores using DDS Entropy,

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
DDS Entropy	7.80E-05	1.01E-02	1.19E-03	1.09E-05	-	-	1.13E-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 2: *The average in situ execution times, in seconds, for each metric to evaluate a single viewpoint of the Lulesh simulation. The pre-processing stage of the workflow 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 executed for each considered camera placement.*

- Figure 3 shows heatmaps for 10 VQ metrics for a single data set, and
- Figure 4 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 4, 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 4, 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. This is a fortunate property, and contrasts with several of the score heatmaps in Figure 3, which have local optimums that may be hard to locate quickly. Overall, we find these heatmaps to be promising with respect to the searchability of the space.

4.2.2. Search Algorithm Evaluation

This evaluation considered all algorithms from Section 3.2. We tested our search algorithms on the heatmaps from Section 4.2.1, evaluating how many camera positions each search algorithm needs to evaluate in order to find a quality viewpoint.

Table 3 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. While the primary takeaway from this table is on efficacy of the algorithms, another takeaway is the large number of searches needed to find a good view. Fortunately, this number is greatly reduced when considering just Entropy metrics.

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 3: *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 4: *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.*

The results of limiting this analysis to just Entropy metrics are located in Table 4. As a reminder, the Entropy metrics are worthy of special attention, since they are the ones that best predicted user preference [M*21]. most closely align with user preference [M*21]. Diagonal 43 was once again the best performer, although the number of views considered was considerably less: only 2.1 viewpoints were needed to find an 80^{th} percentile view, which represents a 10X reduction in the number of searches compared to

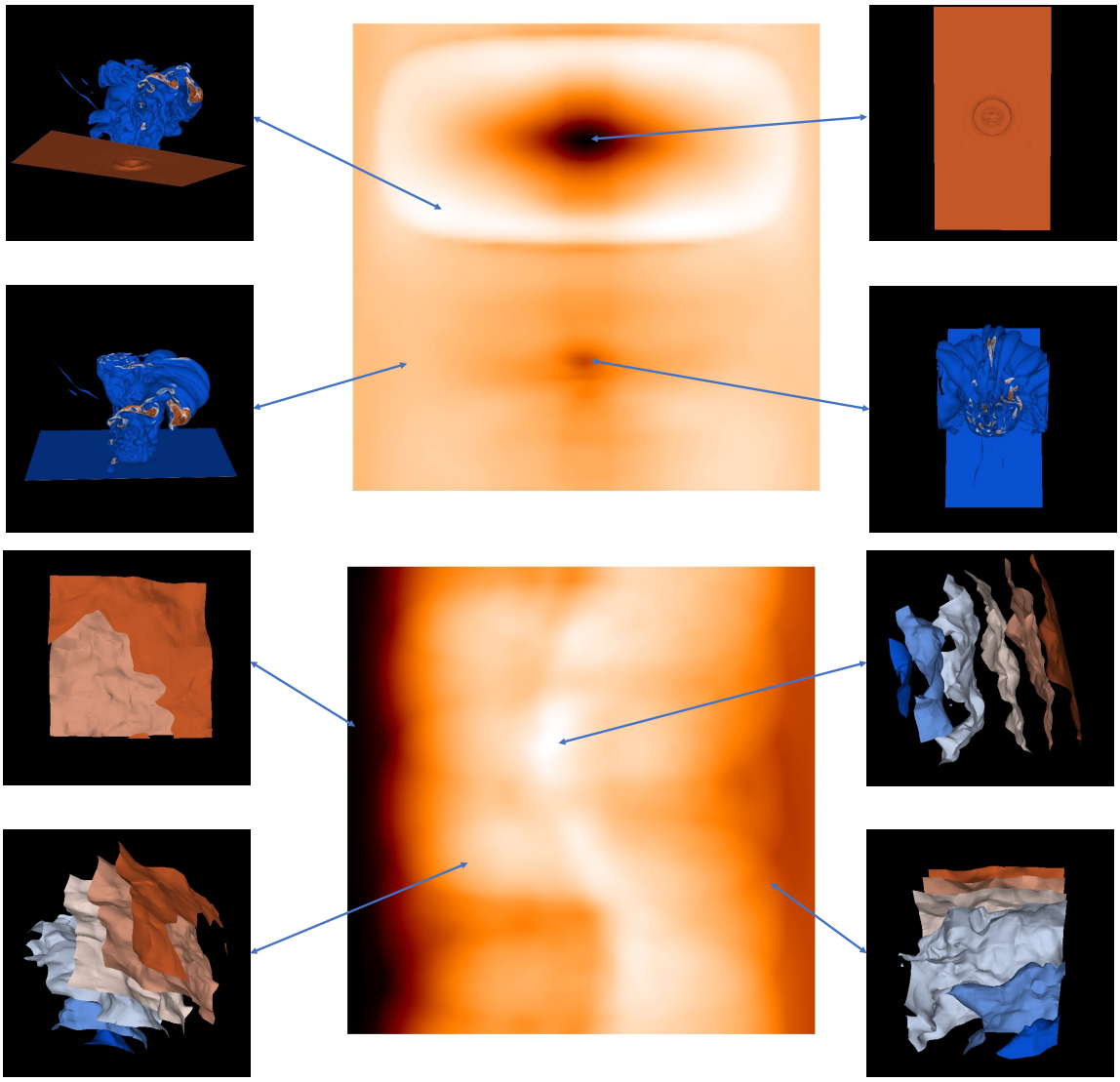

Figure 2: *Annotated heatmaps of the Asteroid (top) and Constit (bottom) data sets. For the Asteroid data set, the DDS Entropy scores from best to worst are: top left, bottom left, bottom right, top right. For the Constit data set, the best to worst are: top right, bottom left, bottom right, top left.*

the analysis using all VQ metrics. 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 has a 10% better VQ metric score (and keeping in mind that users often disagreed when it came to images with high scores).

4.3. Phase 3: In Situ Evaluation

This phase builds on the previous two phases by performing an situ evaluation. Section 4.1 demonstrated that parallel VQ metric calculation can be executed in situ, adding little overhead to the more costly rendering process. Section 4.2 provided insight into best practices when searching for a camera placement.

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. 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

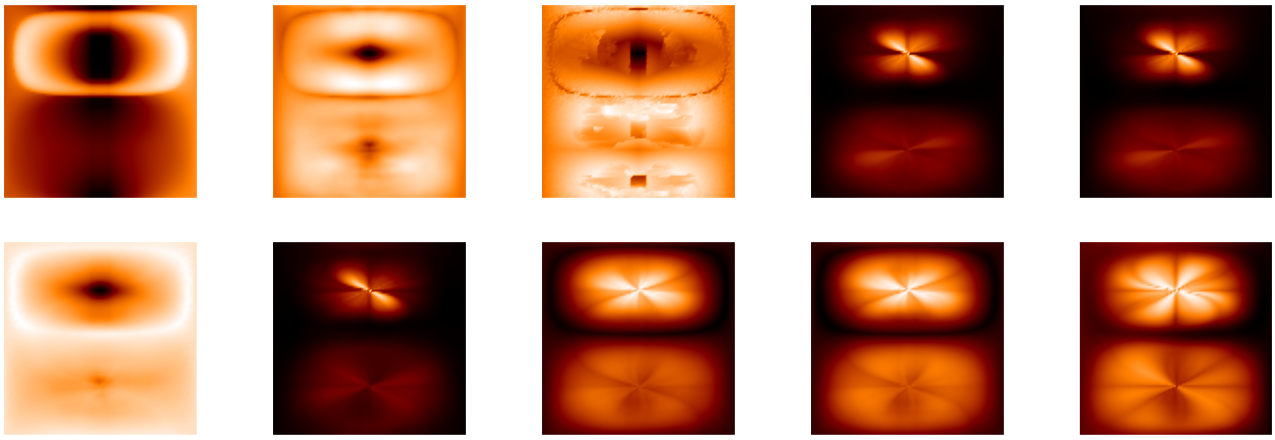

Figure 3: *Ten heatmaps corresponding to ten VQ metrics on the Asteroid data set. The VQ metrics are, from top left to bottom right: Data Entropy, Depth Entropy, Max Depth, PB, Projected Area, Shading Entropy, Viewpoint Entropy, Visibility Ratio, Visible Triangles, VKL. The Asteroid heatmap for DDS Entropy can be found in Figure 4. These heatmaps show that the searchability of the space is dependent on VQ metric.*

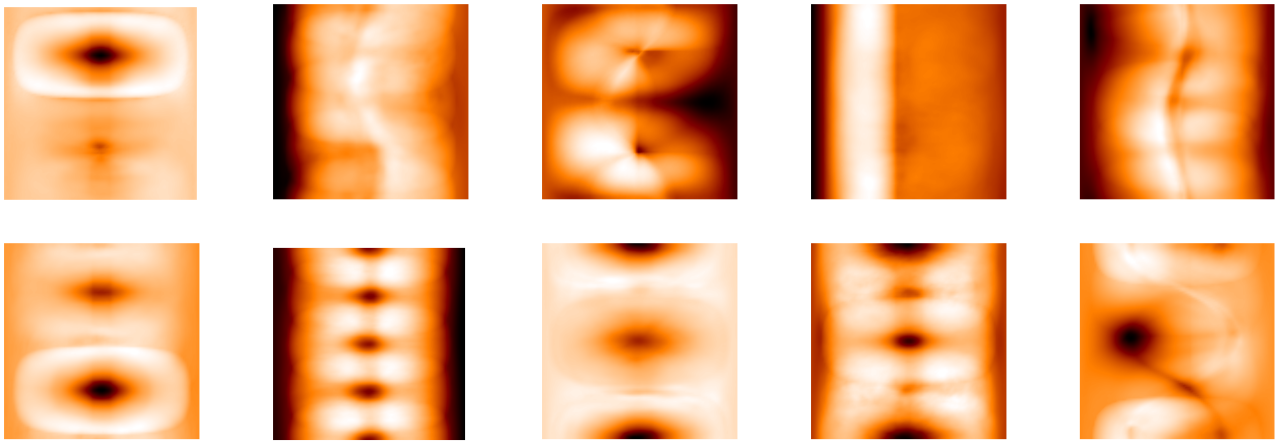

Figure 4: *Heatmaps for the DDS Entropy metric for ten data sets. The data sets are, from top left to bottom right: Asteroid, Constit, ExaAm Truchas, Fluid Dynamics, Hurricane, Mantle, Miranda, S3D-N2, S3D-UVEL, ExaSky Nyx. These heatmaps show that the higher scores or "hot spots" are not sparse nor randomly appearing.*

different budgets: 5, 10, 20, 50 and 100 camera placements. The five budgets control how many camera placements are evaluated per cycle. Given a camera budget, the camera placements were chosen using Diagonal 43, i.e., the best performing search algorithm from Section 4.2.2, Note that our search algorithm starts from the same location for each camera budget, meaning that the camera placements for a smaller budget are a subset of the camera 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 5 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

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 5: *Metric scores and execution times for all five budgets and all VQ metrics for our in situ evaluation using the Diagonal 43 search algorithm with the Lulesh simulation. 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 of the Lulesh simulation, which is representative.*

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 6: *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.*

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 across all budgets. Once normalized, we calculate the percentage change in metric score when increasing the camera budget, and then took the average across all cycles. Table 6 shows the average increase in VQ metric score for each camera budget.

On average, few of the metrics showed any substantial increase in metric score when increasing the camera budget. Depth Entropy and Max Depth had the most substantial percentage increase, each experiencing an average increase in score of 11%, when increasing the number of evaluated camera positions from 20 to 50. Depth Entropy also had an 8% increase in score when the number of evaluated camera positions increased from 50 to 100, but requires more than doubling 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 evaluated camera positions, 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 3 and Table 3 would sug-

gest for individual VQ metrics. The percentage increases, or lack thereof, reinforces the conclusions from Section 4.2: that, depending on the VQ metric, the search algorithms can find a viewpoint with a reasonably high VQ metric score quite quickly, but finding the "best" viewpoint with the highest metric score may take longer than in situ constraints allow.

5. Conclusion

The goal of this research was to establish the viability of automatic in situ camera placement, i.e., 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 [M*21] 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 is supported by 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.

Immediate future work is two-fold: (1) examining the behavior of optimal camera placement over time, (2) design and evaluate best practices for how often a search should be conducted. In regards to the first 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. Of note, the time-varying version of this problem can amortize the burden of search times over simulation cycles — a search can happen for one cycle, and then a good camera can be re-used for many cycles before a new search is needed. More broadly, future work could explore and evaluate other search algorithms and VQ metrics. Moreover, this work could be extended to consider other data sets and rendering techniques, such as volume rendering, and multivariate data. Other ideas include evaluating a camera pathline and saving a time-dependent or -independent movie, or saving a spectrum of quality images. Finally, our spherical coordinates-based search oversamples at the poles, and there are better approaches for equally sampling the space [L*18]. Incorporating such an approach should be adopted in future work.

6. Acknowledgments

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.

7. Appendix

Listing 2: *C++ code of the templated struct MaxValueWithChecks that uses VTK-m for determining a valid maximum value.*

```
template <typename T>
struct MaxValueWithChecks
{
  MaxValueWithChecks(T minValid, T maxValid)
      : MinValid(minValid),
        MaxValid(maxValid) {}
  VTKM_EXEC_CONT inline T operator()(const T
      &a, const T &b) const
  {
    if (this->IsValid(a) && this->IsValid(b)
        ) { return (a > b) ? a : b;}
    else if (!this->IsValid(a)) { return b;}
    else if (!this->IsValid(b)) { return a;}
    else { return this->MinValid;}
  }
  VTKM_EXEC_CONT inline bool IsValid(const T
      &t) const
  {
    return !vtkm::IsNan(t) && t > MinValid
       && t < MaxValid;
  }
  T MinValid;
  T MaxValid;
};
```

References

[A*13] ALMGREN A. S., ET AL.: Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal 765*, 1 (2013), 39. 5

[A*14] AHRENS J., ET AL.: An image-based approach to extreme scale in situ visualization and analysis. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014), IEEE, pp. 424–434. 4

[B*11] BONAVENTURA X., ET AL.: Viewpoint information. *21st International Conference on Computer Graphics and Vision, GraphiCon'2011 - Conference Proceedings* (2011). 2

[B*18] BONAVENTURA X., ET AL.: A survey of viewpoint selection methods for polygonal models. *Entropy 20*, 5 (2018). 2

[B*19] BELAK J., ET AL.: Exaam: Additive manufacturing process modeling at the fidelity of the microstructure. In *APS March Meeting Abstracts* (2019), vol. 2019, pp. C22–010. 5

[Bla87] BLAHUT R. E.: *Principles and Practice of Information Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987. 2

[BR82] BURBEA J., RAO C.: On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory 28*, 3 (1982), 489–495. 2

[BS05] BORDOLOI U., SHEN H.: View selection for volume rendering. In *16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, October 23-28, 2005* (2005), pp. 487–494. 2

[C*05] COOK A. W., ET AL.: 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. 5

[C*19] CARSON R. A., ET AL.: ExaConstit, 2019. 5

[CM09] CORREA C. D., MA K.-L.: Visibility-driven transfer functions. In *2009 IEEE Pacific Visualization Symposium* (2009), pp. 177–184. 2

[D*10] DUTAĞACÄŚ H., ET AL.: A benchmark for best view selection of 3d objects. *3DOR'10 - Proceedings of the 2010 ACM Workshop on 3D Object Retrieval, Co-located with ACM Multimedia 2010* (2010), 45–50. 2

[F*09] FEIXAS M., ET AL.: A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Trans. Appl. Percept. 6*, 1 (2009), 1:1–1:23. 2

[I*98] ITTI L., ET AL.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (1998), 1254–1259. 2

[JS06] JI G., SHEN H.-W.: Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1109–1116. 2

[K*04] KUO B., ET AL.: Hurricane ISABEL simulation data. http://vis.computer.org/vis2004contest/data.html, 2004. 5

[Kuh14] KUHNERT J.: Meshfree numerical schemes for time dependent problems in fluid and continuum mechanics. *Advances in PDE modeling and computation* (2014), 119–136. 5

[L*05] LEE C. H., ET AL.: Mesh saliency. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 659–666. 2

[L*11] LEE T.-Y., ET AL.: View point evaluation and streamline filtering for flow visualization. In *IEEE Pacific Visualization Symposium* (2011), pp. 83–90. 2

[L*17] LARSEN M., ET AL.: 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. 3

[L*18] LUKASCZYK J., ET AL.: Voidga: A view-approximation oriented image database generation approach. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)* (2018), pp. 12–22. 10

[Lin91] LIN J.: Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory 37*, 1 (1991), 145–151. 2

[M*16] MORELAND K., ET AL.: VTK-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications 36*, 3 (2016), 48–58. 3

[M*21] MARSAGLIA N., ET AL.: An Entropy-Based Approach for Identifying User-Preferred Camera Positions. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)* (2021). 2, 3, 5, 6, 10

[N*07] NARAOKA R., ET AL.: Locating an optimal light source for volume rendering. *Journal of the Visualization Society of Japan 27* (2007), 45–46. 2

[P*03] PAGE D., ET AL.: Shape analysis algorithm based on information theory. In *IEEE International Conference on Image Processing* (2003), vol. 1, pp. 229–232. 2

[P*05] POLONSKY O., ET AL.: What's in an image? *The Visual Computer 21*, 8 (Sep 2005), 840–847. 2

[PB96] PLEMENOS D., BENAYADA M.: Intelligent display in scene modelling. new techniques to automatically compute good views. In *GraphiCon'96* (Saint Petersburg (Russia), 1996). 2, 3

[PG17] PATCHETT J., GISLER G.: Deep water impact ensemble data set. https://sciviscontest2018.org/wp-content/uploads/sites/19/2017/09/DeepWaterImpactEnsembleDataSet_Revision1.pdf. 5

[Ple91] PLEMENOS D.: *Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes : le projet multiformes*. PhD thesis, University of Nantes, 1991. 2

[S*05] SBERT M., ET AL.: 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. 2, 3

[S*11a] SECORD A., ET AL.: Perceptual models of viewpoint preference. *ACM Trans. Graph. 30*, 5 (2011), 109:1–109:12. 2, 3

[S*11b] SHAHNAS M. H., ET AL.: The high-pressure electronic spin transition in iron: Potential impacts upon mantle mixing. *Journal of Geophysical Research: Solid Earth 116*, B8 (2011). 5

[SFR*02] SBERT M., FEIXAS M., RIGAU J., VIOLA I., CHOVER M.: Applications of information theory to computer graphics. In *Eurographics* (2002). 2

[SP05] SOKOLOV D., PLEMENOS D.: Viewpoint quality and scene understanding. In *Proceedings of the 6th International Conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage* (Aire-la-Ville, Switzerland, Switzerland, 2005), VAST'05, Eurographics Association, pp. 67–73. 2

[SS02] STOEV S. L., STRASSER W.: A case study on automatic camera placement and motion for visualizing historical data. In *IEEE Visualization, 2002. VIS 2002.* (2002), pp. 545–548. 2

[T*05a] TAKAHASHI S., ET AL.: A feature-driven approach to locating optimal viewpoints for volume visualization. In *VIS 05. IEEE Visualization, 2005.* (2005), pp. 495–502. 2

[T*05b] TAKAHASHI S., ET AL.: Interval volume decomposer: a topological approach to volume traversal. In *Visualization and Data Analysis 2005* (2005), Erbacher R. F., Roberts J. C., Grohn M. T., Borner K., (Eds.), vol. 5669, International Society for Optics and Photonics, SPIE, pp. 103 – 114. 2

[T*09] TAO Y., ET AL.: Structure-aware viewpoint selection for volume visualization. In *2009 IEEE Pacific Visualization Symposium* (2009), pp. 193–200. 2

[T*13] TAO J., ET AL.: 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. 2

[T*16] TAO Y., ET AL.: Similarity voting based viewpoint selection for volumes. *Computer Graphics Forum 35* (2016). 2

[T*17] TREICHLER S., ET AL.: S3d-legion: An exascale software for direct numerical simulation of turbulent combustion with complex multicomponent chemistry. 5

[V*01] VÁZQUEZ P.-P., ET AL.: Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), VMV '01, Aka GmbH, pp. 273–280. 2, 3

[V*04] VIOLA I., ET AL.: Importance-driven volume rendering. In *IEEE Visualization 2004* (2004), pp. 139–145. 2

[V*06] VIOLA I., ET AL.: Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 933–940. 2

[V*09] VIEIRA T., ET AL.: Learning good views through intelligent galleries. *Comput. Graph. Forum 28* (2009), 717–726. 2

[XY15] XIAOLING Y., YANNI W.: A view selection method based on particle swarm optimization. In *2015 International Conference on Computers, Communications, and Systems (ICCCS)* (2015), pp. 69–72. 2