LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-840594

A Trigger-Based Approach for Optimizing Camera Placement Over Time

N. Marsaglia, M. Majumder, H. Childs

September 30, 2022

Supercomputing ISAV 2022
Dallas, TX, United States
November 13, 2022 through November 18, 2022

Disclaimer

A Trigger-Based Approach for Optimizing Camera Placement Over Time

Nicole Marsaglia
Lawrence Livermore National Laboratory
Livermore, USA
marsaglia1@llnl.gov

Meghanto Majumder
The University of Oregon
Eugene, USA
meghanto@uoregon.edu

Hank Childs
The University of Oregon
Eugene, USA
hank@uoregon.edu

***Abstract*—We contribute a new approach for in situ automation of camera placement over time. Our approach incorporates triggers, regularly evaluating the current camera placement and searching for a new camera placement when a trigger fires. We evaluate our approach running in situ with five data sets from two simulation codes, considering camera placement quality (evaluated using a viewpoint quality metric) and overhead (number of camera positions evaluated). We find that our approach has a significant benefit — reduced overhead with similar quality — compared to the naïve approach of searching for a new camera placement each cycle.**

***Index Terms*—In Situ Visualization, Automated Visualization**

I. Introduction

One important aspect of creating a scientific visualization is setting the camera position to a location that enables insight. With post hoc visualization, camera positions are typically set by scientists — a visualization program starts with an initial camera position and then scientists can iterate until locating a suitable camera position. Importantly, this workflow is possible because post hoc visualization often occurs in a human-in-the-loop (HITL) setting. With in situ visualization, however, the workflow often is not HITL, meaning that camera positions must be selected in some other way. One approach is to identify a suitable camera position a priori, typically by using a predecessor calculation that can be visualized post hoc or by exploiting knowledge from a domain scientist. Another approach, used by the Cinema [1] project, is to render from many camera positions, in hopes that at least one will be useful. With this work, we consider a third approach: automating the selection of a camera position while the simulation runs. This approach has the potential to find good camera positions with reduced calculation and no a priori knowledge.

This paper builds on two recent results that together provide a viable in situ approach for finding a camera position at a single time step. First, Marsaglia et al. [27] considered how Viewpoint Quality (VQ) metrics can predict user preference for isosurface data. Their contribution was two-fold: (1) a new VQ metric, "DDS Entropy," which combined data entropy, depth entropy, and shading entropy, and (2) a user study on isosurface data that established their new VQ metric is a good rough predictor of preferred views, i.e., when comparing two camera positions, users often prefer the position with the higher DDS Entropy score. Second, Marsaglia et al. [26] then demonstrated how this approach could be used in an in situ setting. Once again, they made a two-fold contribution: (1) an algorithm to efficiently calculate DDS Entropy in a distributed-memory parallel setting, and (2) an approach for efficiently searching through the space of possible camera positions to identify a good camera position.

An important consideration for their approach is the execution time. Evaluating a VQ metric at a given camera position requires rendering the scene as well as performing some additional calculations, meaning each VQ metric evaluation is slightly slower than rendering a scene. Further, their search approach incorporated a "budget," i.e., a budget of N would consider N possible camera positions, with the search algorithm intelligently choosing candidate positions to maximize metric score. That said, Marsaglia et al. recommended lower search budgets, on the order of ten positions overall, since users often disagree what view is "best" after a sufficiently good view is found. In all, a search with a budget of ten would take about the time it takes to render scenes from ten camera positions. Further, in the context of a simulation evolving in time, a trivial extension of Marsaglia et al.'s works would be to apply their approach at every cycle (or at least each cycle where visualization is performed). This extension would succeed in finding good views, but it would perform many unnecessary calculations, as it would be doing a fresh search at each cycle.

With this study, we extend Marsaglia et al.'s contributions to a time-varying setting. The main goal of our approach is to achieve good camera positions with less work than the trivial extension. We do this with a trigger-based approach where the view from the previous time step is reused unless a trigger "fires" indicating a new search should take place. Our contributions are two-fold: (1) a proposed approach for time-varying camera placement and (2) an evaluation of how that approach performs in multiple settings.

II. Related Work

This related work section first discusses the main area of camera placement over time (II-A). It then discusses "triggers" (II-B), which are a key idea in our proposed approach.

A. Camera Placement Over Time

There have been several non-in situ works considering camera placement. First, several works [20], [23], [24], [33] have specifically considered flow visualization. Other works have focused on a path moving through a time-varying data set [4], [16].

With respect to in situ processing, many works have considered computational steering [3], [5], [7]–[15], [28], [29], [31], [35], gaining the benefit of HITL (including camera placements), but incurring other costs. That said, Yamamoto et al. [36] employed an in situ approach that differed from computational steering via HITL. In their approach, they output videos of the simulation as it evolves over time from a number of different viewpoints. These videos were passed to a "Membrane Layer," where they could be inspected by domain scientists. Our approach differs from the Yamamoto et al. approach and from the computational steering approach in that the domain scientist does not need to be involved.

B. Triggers

Triggers are a mechanism for determining when to perform actions and what actions to perform. Ideally, triggers are lightweight enough that they can be run often, and they "fire" only when heavyweight analysis is needed. The in situ community has made heavy use of both domain-agnostic triggers [19], [21], [37] and domain-specific triggers [6], [22], [30], [32], [34], [38]. That said, none of these trigger-based works have considered the problem of camera placement.

III. Our Method

Algorithm 1 Automatic Camera Placement Over Time (ACPOT) Algorithm

```
global C*;
global DDS*;
global searchBudget;
global threshold;
global needsInitialization = true;
function TimeVaryingCameraSelection:
if needsInitialization then
    C* = SearchForCameraPlacement(budget);
    DDS* = EvaluateDDSEntropy(C*);
    needsInitialization = false;
else
    DDS^cur = EvaluateDDSEntropy(C*);
    if ((1+threshold)×DDS* < DDS^cur) or
    (DDS^cur < (1-threshold) ×DDS*) then
        C* = SearchForCameraPlacement(budget);
        DDS* = EvaluateDDSEntropy(C*);
    end if
end if
```

Our method employs a trigger-based approach. We begin by running the Marsaglia et al. search algorithm [26] for finding a good camera placement. We refer to this camera position as $C^{\star}$. We also record the DDS Entropy score for $C^{\star}$ and refer to this score as $DDS^{\star}$. For each subsequent cycle, we evaluate the DDS Entropy score at $C^{\star}$, and refer to this score as DDS^{cur}. This DDS^{cur} value is used for the trigger. If the value has changed significantly from $DDS^{\star}$, then the trigger "fires" and we search for a new view (again using the Marsaglia et al. search algorithm). The resulting view becomes the new $C\star$ and its DDS Entropy score becomes the new $DDS^{\star}$. An important aspect to this approach is how much change is needed to cue a fire. We represent this as a percentage threshold, and our results consider the effect of different threshold settings. Finally, our trigger fires when DDS^{cur} value goes up or down. Decreases in DDS^{cur} can correspond to occlusions that have developed, while increases can indicate that the simulation has evolved (and thus other viewpoints may now be better). Listing 1 shows pseudocode for our algorithm.

IV. Background for Experiments

A. Data Sets

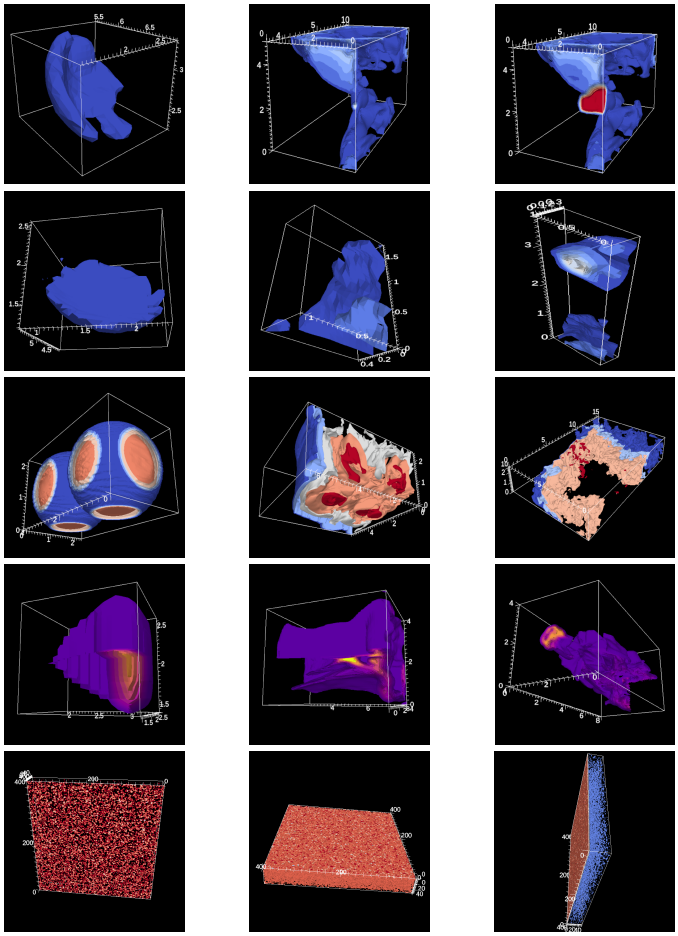

Fig. 1. Renderings of data used in this study. Each row consists of a single data set: IMPACT-ENERGY (top row), IMPACT-PRESSURE, BALL-OF-FURY, JETBOX, AMR-WIND (bottom row). The columns correspond to evolution in time, with the left column being early in the simulation and the right column being late in the simulation. The camera placement for each image is the one that maximizes DDS Entropy score.

We conducted our experiments using two different simulation codes, AMR-Wind [2] and CloverLeaf3D [25], and five total data sets:

- IMPACT-ENERGY: Cloverleaf3D simulation code using the "Impact" input deck and visualizing isosurfaces of the energy field. This simulation was executed for 830 cycles on a 64^3 mesh and a new camera search was performed every ten cycles. The isolevels for the visualization were chosen logarithmically, and one quadrant was clipped away.
- IMPACT-PRESSURE: Identical to CLOVER-IMPACT-ENERGY, except the pressure field was visualized.
- BALL-OF-FURY: Cloverleaf3D simulation code using the "Ball of Fury" input deck and visualizing isosurfaces of the energy field. This simulation was executed for 9100 cycles on a 64^3 mesh and a new camera search was performed every hundred cycles. The isolevels for the visualization were chosen logarithmically.
- JETBOX: Cloverleaf3D simulation code using the "Jetbox" input deck and visualizing isosurfaces of the energy field. This simulation was executed for 1200 cycles on a $256 \times 128 \times 256$ mesh and a new camera search was performed every ten cycles. The visualization clipped one quadrant away.
- AMR-WIND: AMR-Wind simulation of 65 cycles on a 848^3 mesh, with a new camera search every cycle. The visualization was of isosurfaces.

Figure 1 shows renderings from each data set.

B. Metrics

There are two main ways to evaluate an ACPOT algorithm.

First, how much work is needed to execute the algorithm? Our metric for this is the number of camera placements considered. We use this metric since execution time varies significantly across data set — the time it takes to evaluate the DDS Entropy score for a camera placement is essentially the time it takes to render the data set, and this time varies based on scale, data size, etc.

Second, what is the quality of camera placement achieved by the algorithm? Our metric for this is the loss in DDS Entropy. For example, if a fresh search found camera $C^\star$ with DDS Entropy score $DDS^\star$ and if the ACPOT algorithm recommended camera C^{cur} with score DDS^{cur}, then loss in DDS Entropy would be $DDS^\star - DDS^{cur}$. We chose this metric because of the previous user study by Marsaglia et al. which that demonstrated that users prefer higher DDS Entropy scores. That said, their previous work only established that a higher DDS Entropy score led to increased user preference, and did not provide intuition about what magnitude change in metric value is significant. For this study, we have reanalyzed their user study corpus to consider the impact of changes in DDS Entropy. This new analysis is in Table I. From this analysis, we conclude that searching for views where the DDS Entropy improvement is less than 0.25 is unlikely to be beneficial (i.e., result in finding a view that a user prefers over the current one), but searching for views where the improvement is 0.5 or bigger is more likely to be beneficial (with 0.75 and up particularly so).

TABLE I
RE-ANALYZING THE USER STUDY FROM MARSAGLIA ET AL. TO INFORM WHAT MAGNITUDE DIFFERENCE IN DDS ENTROPY IS SIGNIFICANT. THEIR USER STUDY CONSIDERED PAIRS OF CAMERAS AND ASKED USERS TO CHOOSE THEIR PREFERENCE BETWEEN THEM. IF THE DIFFERENCE IN DDS ENTROPY BETWEEN TWO CAMERA PLACEMENTS WAS BETWEEN 0 AND 0.25, THEN USERS PREFERRED THE CAMERA WITH THE HIGHER DDS ENTROPY SCORE ONLY 55% OF THE TIME. HOWEVER, IF THE DIFFERENCE BETWEEN THEM WAS GREATER THAN 1.0, THEN USERS PREFERRED THE CAMERA WITH THE HIGHER DDS ENTROPY SCORE 79% OF THE TIME.

Difference in Entropy	0-0.25	0.25-0.5	0.5-0.75	0.75-1.0	1.0+
User Preference	55%	58%	62%	80%	79%

V. RESULTS

This section details our results in two phases. The two main "knobs" in our algorithm are the search budget and threshold. The first phase considers search budget, i.e., how much does VQ metric score improve when the search budget increases? This phase is conducted entirely through post hoc experiments. The second phase uses the findings from the first (specifically narrowing the search budget to options of 10 or 20) and then considers how our ACPOT algorithm behaves overall. This phase is conducted entirely through in situ experiments.

A. Phase 1: Search Budget

As discussed in the introduction, evaluating a VQ metric for a given camera position has non-negligible cost. As a result, it is desirable for an ACPOT algorithm to produce good views with a "small" number of VQ metric evaluations. In all, the goal of this phase is to inform how many views should be considered when doing a search (i.e., search budget).

Our experiments used the DDS Entropy metric on the IMPACT-ENERGY, IMPACT-PRESSURE, and BALL-OF-FURY data sets. We considered search budgets of 1, 4, 5, 10, 20, 25, and 100. For some of our analyses, 100 was used as a reference, e.g., how did a search with N camera positions compare to the search with 100 camera positions? The camera placements were chosen using the Fibonacci Lattice, a method for equally distributing points around a sphere.

Selecting a subset of camera positions can lead to noisy data. For example, for a search budget of four, choosing positions $[0, 25, 50, 75]$ on the Fibonacci lattice may lead significantly different results than choosing positions $[12, 37, 62, 87]$. To counteract this, we averaged across all possible "evenly-spaced" selections for a budget. For example, for a budget of four, we considered $[0, 25, 50, 75], [1, 26, 51, 75], \ldots, [24, 49, 74, 99]$, and used the average result across the 25 possible selections.

Figure 2 shows the results of the tradeoff between the search budget and average obtained VQ metric score (i.e., DDS Entropy). Considering just a single view (i.e., search budget equals one) is clearly a bad strategy, which is not surprising. Further, budgets of four and five show some noticeable deviation from the larger budgets. On the other end,

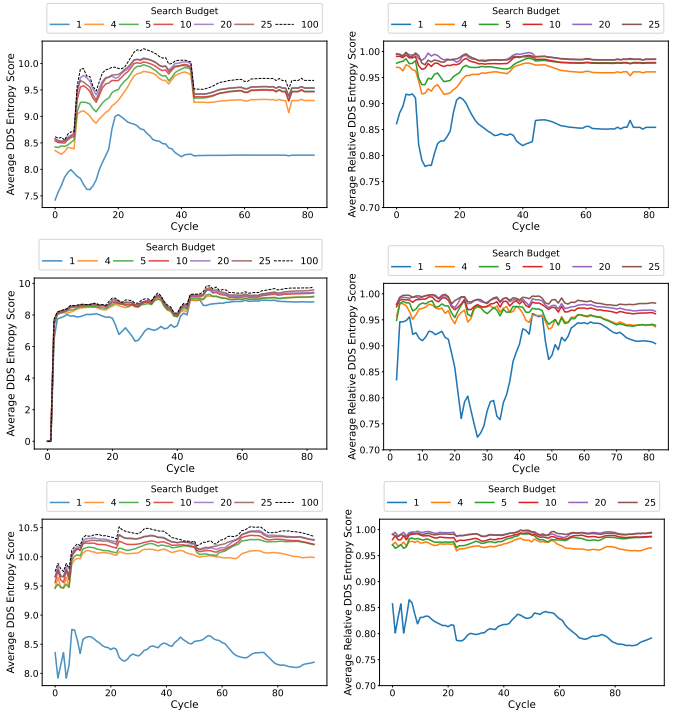

Fig. 2. Six figures considering the effect of search budget. Each of the six figures shows the evolution over time, i.e., the simulation cycle number is the X-Axis of the plot. The Y-Axis varies based on analysis. In the left column, the analysis shows the absolute scores for DDS Entropy. In the right column, the analysis considers results normalized by the DDS Entropy score from a search budget of 100. Finally, each row corresponds to a data set: IMPACT-ENERGY (top), IMPACT-PRESSURE (middle), and BALL-OF-FURY (bottom).

a budget of 25 only added benefit for the IMPACT-PRESSURE data set, and was similar to a budget of 10 or 20 for other data sets. In all, we decided to focus our Phase 2 results on search budgets of 10 and 20, as they provide much of the quality compared to a search budget of 100 views, but at reduced cost.

B. Phase 2: ACPOT Performance

This phase evaluates the performance of the ACPOT approach. This phase considers the remaining "knob" — threshold — and then considers the performance of the algorithm with respect to the two metrics discussed in Section IV-B: number of camera positions considered and loss in DDS Entropy score.

The experiments for this phase are as follows. First, the data sets considered were BALL-OF-FURY, JETBOX, and AMR-WIND. All experiments were performed in situ with the simulation data being passed to Ascent [17], an in situ analysis and visualization library, where it is evaluated by our algorithm. The Cloverleaf3D runs occurred on our local cluster ("Alaska") on a single node. The AMR-Wind runs occurred on the Summit supercomputer at Oak Ridge on 25 nodes using 130 MPI ranks and 130 GPUs. Finally, we considered six threshold percentages: 0%, 1%, 2%, 3%, 4%, and 5%. Note that a 0% threshold means the trigger for the ACPOT algorithm will be triggered every time the DDS Entropy score changes, which in effect means that a new search will happen every cycle.

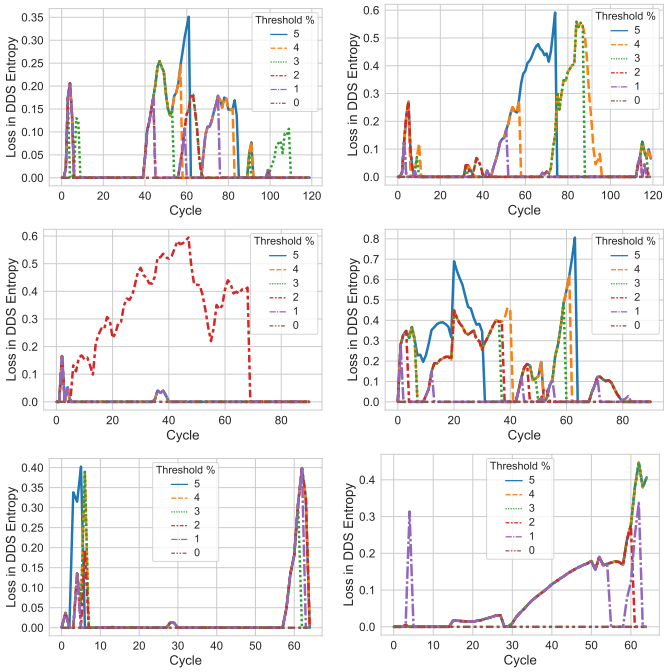

Fig. 3. Six figures plotting the DDS Entropy loss with respect to threshold. Each of the six figures shows the evolution over time, i.e., the simulation cycle number is the X-Axis of the plot. The Y-Axis for each is the loss in DDS Entropy score from using our ACPOT algorithm compared to the strategy of searching for a new camera each cycle. The rows are organized by data set: JETBOX (top), BALL-OF-FURY (middle), and AMR-WIND (bottom). The columns are organized by search budget: 10 (left) and 20 (right).

Figure 3 and Table II describe both the effect of threshold and the efficacy of our ACPOT algorithm. Figure 3, which plots how each threshold performs as the simulation evolves, shows that our approach rarely results in rising above 0.5 (the value we identified as one where we should be searching for new views), and for the most part the views that our ACPOT algorithm returns stays beneath that value. That said, Figure 3 shows the BALL-OF-FURY data set with a search budget of ten shows an issue where the search resulted in a view that did not change DDS Entropy score as the simulation evolved. As a result, it got "stuck" in a bad view. This failure, which can be seen in our results in this section as well, suggests future work (see Section VI). Table II shows the time-averaged results for each of the curves in Figure 3. The average loss in DDS Entropy is small, but it is important to keep in mind that these are averages, and (as seen in Figure 3) these averages represent some intervals in time where it would likely be worthwhile to search for a new camera position. Table II also shows that the speedup from our ACPOT algorithm is significant when compared to the strategy of searching for a new camera placement each cycle — as much as 10X faster for some configurations. Overall, Table II provides significant evidence that our algorithm is achieving its goals — a significant speedup with only a small loss in quality. Further, higher thresholds (like 5%) seem to perform

fairly well. Finally, Figure 4 provides additional analysis on the relationship between overhead and quality. In particular, once the number of trigger "fires" drops below a threshold, the DDS Entropy loss starts going up quite quickly.

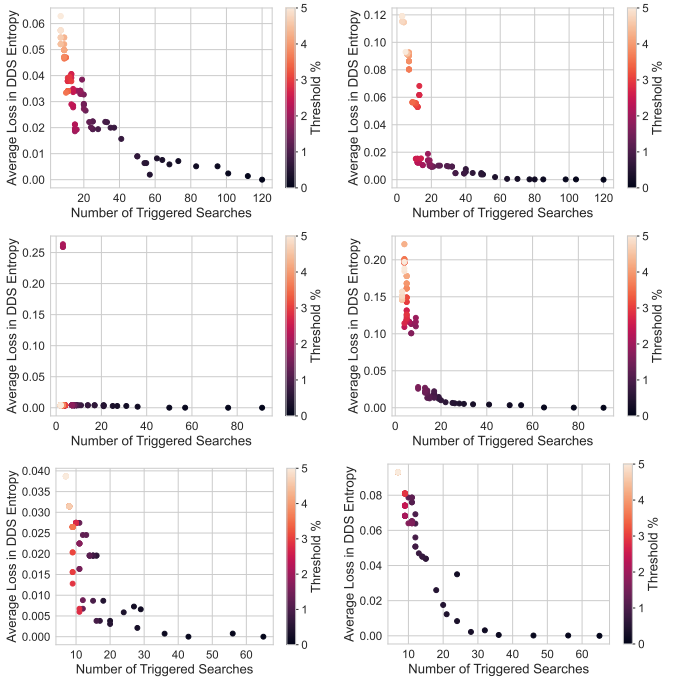

Fig. 4. Six figures comparing the relationship between the number of triggered searches and DDS Entropy loss given a threshold percentage. Each figure considers 100 threshold values (ranging from 0% to 5%) and makes a scatter plot out of the 100 results. The X-Axis is the number of times a trigger fires and generates a new search. The Y-Axis is the average DDS Entropy loss compared to the strategy of searching for a new camera each cycle. The rows are organized by data set: JETBOX (top), BALL-OF-FURY (middle), and AMR-WIND (bottom). The columns are organized by search budget: 10 (left) and 20 (right). The columns on the right have higher DDS Entropy loss in some cases because they are comparing to a higher standard (the best out of 20 possible camera placements instead of 10).

VI. Conclusion

Our ACPOT approach provides a new option for automated camera placement. Two important premises of our research are that (1) many simulation teams do not have a priori knowledge of where to place the camera, especially in the context of evolving simulations and (2) the available time for visualization for some simulation campaigns are small enough that it is not possible to render a bunch of camera positions (e.g., Cinema). When both of these premises are true, we feel our ACPOT approach is currently the best available option, especially in light of our experimental results.

That said, there are many opportunities for future work. First, our algorithm failed by getting "stuck" in one configuration, and we believe our algorithm can be improved by searching for new views at regular durations. Second, DDS Entropy is an imperfect predictor of user preference. Further, it has only been shown to be effective for isosurfaces. That said, as new VQ metrics become available, we feel our ACPOT approach can be adapted in a straight-forward way to incorporate these metrics. Finally, while we evaluated an appreciable number of data sets, search budgets, and thresholds, performing additional evaluation would further demonstrate our approach's efficacy. We plan to do this by running with additional code teams. This should be straight forward to accomplish, since our approach has been deployed in Ascent [18].

TABLE II

The Phase 2 results of applying our ACPOT algorithm in situ for the three datasets: Jetbox, Ball of Fury, and AMR-Wind. For each simulation run, the ACPOT algorithm utilizes a trigger based on a percentage threshold (% Th), as well as a fixed camera budget (CB). This table shows the number of times the trigger was fired (# of T) over the simulation run, the total number of camera evaluations (# of CE), the average loss in DDS Entropy score (ALDE) between the VQ metric score of the selected camera and the VQ metric score of the best possible camera amongst the camera budget, as well as the overall speedup (SUP) achieved when using our ACPOT algorithm compared to the trivial scenario of executing a new camera search ever cycle.

Dataset	% Th	CB	# of T	# of CE	ALDE	SUP
Jetbox	5%	10	7	192	.06	6.3X
	4%	10	9	210	.05	5.7X
	3%	10	11	228	.04	5.3X
	2%	10	16	273	.02	4.4X
	1%	10	26	363	.02	3.3X
	0%	10	120	1200	0.0	0.0X
	5%	20	5	234	.09	10.3X
	4%	20	7	272	.09	8.8X
	3%	20	11	348	.05	6.9X
	2%	20	15	424	.01	5.7X
	1%	20	32	747	.01	3.2X
	0%	20	120	2400	0.0	0.0X
Ball of Fury	5%	10	2	118	.004	7.7X
	4%	10	2	118	.004	7.7X
	3%	10	4	136	.004	6.7X
	2%	10	3	127	.26	7.2X
	1%	10	9	181	.004	5.0X
	0%	10	91	910	0.0	0.0X
	5%	20	4	186	.19	9.8X
	4%	20	5	205	.18	8.9X
	3%	20	5	186	.14	8.9X
	2%	20	6	224	.11	8.1X
	1%	20	17	433	.01	4.2X
	0%	20	91	1820	0.0	0.0X
AMR-Wind	5%	10	7	137	.04	4.7X
	4%	10	8	146	.03	4.5X
	3%	10	9	155	.02	4.2X
	2%	10	10	164	.03	4.0X
	1%	10	14	200	.02	3.3X
	0%	10	65	650	0.0	0.0X
	5%	20	7	217	.09	6.0X
	4%	20	7	217	.09	6.0X
	3%	20	7	217	.09	6.0X
	2%	20	9	255	.07	5.1X
	1%	20	12	312	.07	4.2X
	0%	20	65	1300	0.0	0.0X

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC (LLNL-CONF-840594). 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.

REFERENCES

[1] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. 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*, pp. 424–434. IEEE, 2014.

[2] S. Ananthan, M. Brazell, J. Rood, G. Vijayakumar, M. Henry de Frahan, A. Almgren, and W. Zhang. Amr-wind, aug 2020.

[3] A. Atanasov, H.-J. Bungartz, J. Frisch, M. Mehl, R.-P. Mundani, E. Rank, and C. van Treeck. Computational steering of complex flow simulations. *High Performance Computing in Science and Engineering, Garching/Munich 2009*, p. 63–74, 2010.

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

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

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

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

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

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

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

[11] G. Eisenhauer, K. Schwan, W. Gu, and N. Mallavarupu. 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*, pp. 26–33, 1994.

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

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

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

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

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

[17] M. Larsen, J. Ahrens, U. Ayachit, E. Brugger, H. Childs, B. Geveci, and C. Harrison. 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*, ISAV'17, pp. 42–46. ACM, New York, NY, USA, 2017.

[18] M. Larsen, E. Brugger, H. Childs, and C. Harrison. Ascent: A Flyweight In Situ Library for Exascale Simulations. In *In Situ Visualization For Computational Science*, pp. 255 – 279. Mathematics and Visualization book series from Springer Publishing, Cham, Switzerland, May 2022.

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

[20] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis. View point evaluation and streamline filtering for flow visualization. In *2011 IEEE Pacific Visualization Symposium*, pp. 83–90, 2011.

[21] J. Ling, W. Kegelmeyer, K. Aditya, H. Kolla, K. Reed, T. Shead, and W. Davis. Using feature importance metrics to detect events of interest in scientific computing applications. pp. 55–63, 10 2017. doi: 10.1109/LDAV.2017.8231851

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

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

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

[25] A. Mallinson, D. A. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. A. Jarvis. Cloverleaf: Preparing hydrodynamics codes for exascale. *The Cray User Group*, 2013, 2013.

[26] N. Marasaglia, M. Mathai, S. Fields, and H. Childs. Automatic In Situ Camera Placement for Large-Scale Scientific Simulations. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pp. 49–59. Rome, Italy, June 2022.

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

[28] S. Parker and C. Johnson. Scirun: A scientific programming environment for computational steering. In *Supercomputing '95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, pp. 52–52, 1995.

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

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

[31] C. Stein, D. Bennett, P. Farrell, and A. Ruttan. A steering and visualization toolkit for distributed applications. pp. 451–457, 01 2006.

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

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

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

[35] J. Wood, K. Brodlie, and J. Walton. gviz - visualization and steering for the grid. 01 2003.

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

[37] Y. Yamaoka, K. Hayashi, N. Sakamoto, and J. Nonaka. 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*, ISAV '19, p. 12–16. Association for Computing Machinery, New York, NY, USA, 2019.

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