Astronomy and Computing 33 (2020) 100424

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

Clustering-informed cinematic astrophysical data visualization with application to the Moon-forming terrestrial synestia

P.D. Aleo [a,b,1,*], S.J. Lock [c], D.J. Cox [b], S.A. Levy [b], J.P. Naiman [b], A.J. Christensen [b], K. Borkiewicz [b], R. Patterson [b]

[a] Department of Astronomy, University of Illinois at Urbana-Champaign, 1002 West Green Street, Urbana, IL 61801, USA
[b] Advanced Visualization Laboratory, National Center for Supercomputing Applications, 1205 West Clark Street, Urbana, IL 61801, USA
[c] Division of Geological and Planetary Sciences, California Institute of Technology, 1200 East California Boulevard, Pasadena, CA 91125, USA

ARTICLE INFO

Article history:
Received 16 July 2020
Accepted 30 August 2020
Available online 9 September 2020

Keywords:
Methods: Data analysis
Methods: Numerical

ABSTRACT

Scientific visualization tools are currently not optimized to create cinematic, production-quality representations of numerical data for the purpose of science communication. In our pipeline `Estra`, we outline a step-by-step process from a raw simulation into a finished render as a way to teach non-experts in the field of visualization how to achieve production-quality outputs on their own. We demonstrate feasibility of using the visual effects software Houdini for cinematic astrophysical data visualization, informed by machine learning clustering algorithms. To demonstrate the capabilities of this pipeline, we used a post-impact, thermally-equilibrated Moon-forming synestia from Lock et al., (2018). Our approach aims to identify "physically interpretable" clusters, where clusters identified in an appropriate phase space (e.g. here we use a temperature–entropy phase–space) correspond to physically meaningful structures within the simulation data. Clustering results can then be used to highlight these structures by informing the color-mapping process in a simplified Houdini software shading network, where dissimilar phase–space clusters are mapped to different color values for easier visual identification. Cluster information can also be used in 3D position space, via Houdini's Scene View, to aid in physical cluster finding, simulation prototyping, and data exploration. Our clustering-based renders are compared to those created by the Advanced Visualization Lab (AVL) team for the full dome show "Imagine the Moon" as proof of concept. With `Estra`, scientists have a tool to create their own production-quality, data-driven visualizations.

1. Introduction

Data visualization, the graphical display of either spatial or temporal data, is a wide field used for data analysis and communication (Borkiewicz et al., 2019a,b; Punzo et al., 2015; Hassan and Fluke, 2011; Barnes and Fluke, 2008; Price, 2007, & references therein). The style and content of data visualization may vary (Borkiewicz et al., 2019a; Goodman, 2012), but there are generally three distinct paradigms:

1. "Information visualization"—typically a two-dimensional representation of relational/non-spatial data via networks, graphs, and charts;
2. "Traditional scientific visualization"—imagery created for three-dimensional spatial data to be analyzed predominantly by scientists for publications in peer-reviewed journals;
3. "Cinematic scientific visualization"—production quality, data-driven imagery with aesthetic appeal designed for mass audiences and large-format screens.

In cinematic scientific visualization, Hollywood techniques of composition, rendering quality, and camera design are chosen to bring about visually attractive presentations of the science. There is supporting evidence such attractive presentations are more educational than unattractive ones (Cawthon and Moere, 2007), and spur interest in scientific topics, even those widely-considered monotonous or difficult-to-learn (Arroio, 2010; Dubeck et al., 1994). Because of cinematic scientific visualization's power to simultaneously entertain, educate, and provide new insight about the science in question, it is important that steps be taken to increase ubiquity and ease of use by non-visualization designers, like domain scientists.

* Corresponding author at: Department of Astronomy, University of Illinois at Urbana-Champaign, 1002 West Green Street, Urbana, IL 61801, USA.
E-mail address: paleo2@illinois.edu (P.D. Aleo).
[1] Fiddler Innovation Fellow.

https://doi.org/10.1016/j.ascom.2020.100424
2213-1337/

By equipping scientists with the means and tools to create cinematic-driven visualizations of their own data, scientists can create new, insightful, broad-reaching imagery for use in all forms of science communication: from peer-reviewed publications, conferences, presentations, and simulation prototyping to public outreach, news media, interviews, and social media (Borkiewicz et al., 2019a; Shih et al., 2019). In an age of misinformation (Borkiewicz et al., 2017), it has never been more important for data-driven visualizations to spearhead scientific communications directly from scientists themselves, even more so than popular science communicators. In fact, it has been shown that user-generated science content via scientists are more popular on Youtube than those created by professional scientific communicators (Welbourne and Grant, 2016).

Currently, the niche field of cinematic scientific visualization is dominated by visualization and visual effects designers who, in communication with scientists, create accurate representations of the data. Scientists, particularly astrophysicists for this work, are largely focused on traditional scientific visualization, designed to be shared and understood by peers exclusively. According to Hassan and Fluke (2011), the astronomical community has put in "limited effort" to develop general purpose, 3D-visualization tools equipped to handle astronomical data. However, recently there has been some development in creating tools for such cinematic visualization. For example: tools implementing astronomical Adaptive Mesh Refinement (AMR) (Berger and Oliger, 1984; Kähler et al., 2002) data into the visual effects software Houdini via `ytini`[2] (Naiman et al., 2017; Borkiewicz et al., 2019b); generating physically-accurate artistic models of astronomical objects with 3D modeling software Blender (Kent, 2013, 2015); combining the analysis tools of `yt` (Turk et al., 2011) with Blender via `AstroBlend` (Naiman, 2016); and applying 3D interactive visualization software to astronomical FITS files via `SlicerAstro` (Punzo et al., 2017). However, tools for scientists to develop cinematic visualizations remain limited and their use is not widespread. With the beginning of the "Petascale Astronomy Era" (Hassan and Fluke, 2011) of next-generation sky surveys and supercomputing facilities, such developments have never been more crucial.

The computational cost of cinematic astrophysical data visualization is decreasing with improving technologies and techniques. As such, there will be a growing interdisciplinary environment for collaborations between scientists and digital artists (Cox, 2008). Visual effects and modeling software like Houdini, Blender, and Unity can read-in scientific data with plug-ins and packages, and are waiting to be more widely-used by the astronomical community. Moreover, some current software is able to provide near-cinematic, real-time rendering by exploiting GPUs, and could be utilized to replicate the results we show in this work. For example, the CUDA implementation of Splotch can visualize large volumes of astronomical, particle-based computer simulations and observations by leveraging GPUs in modern HPC infrastructures (Rivi et al., 2014); Unreal Engine 4 has integrated ray tracing and novel denoising algorithms to achieve cinematic quality real-time rendering (Liu et al., 2019); and GPU-accelerated plug-ins like OctaneRender in Unity[3] enables cinematic-quality path-traced rendering. This work will help scientists become their own storytellers and develop simple, yet effective cinematic astrophysical data visualizations using machine learning clustering algorithms and the visual effects software Houdini.

Machine learning and deep learning techniques and ideas have become prominent in astronomy because of their innate capability to process and analyze big data (Ball and Brunner, 2010; Pesenson et al., 2010; Burke et al., 2019; Baron, 2019, & references therein). With the substantial amount of data generated from large current and future astronomical surveys and simulations, it is imperative that the field of cinematic astrophysical visualization not be left behind, and take advantage of the cutting edge software and hardware available.

The application of machine learning in visualization is a relatively new and underdeveloped field (Ma, 2007). As we show in this work, one application of machine learning that is well suited for visualization is cluster finding. Machine learning algorithms can be used to discover interesting features and classify clusters, and in scientific studies it is up to the researcher to determine the context. Likewise, a visualization artist has a narrative or educational insight they wish to convey to an audience, and this decision informs what features the visualization will highlight.

Often in visualization, only a small subset of all the data attributes (that is, the span of the n-dimensional dataset) are chosen for the final render due to computational costs, aesthetic quality, and other subjective reasoning such as human perception. The visualization artist will spend weeks implementing a "guess and check" method to see what "looks best" when it comes to lighting and coloring (shading) the simulation and adjusting which variables are mapped to luminance and opacity, etc. However, this can lead to key features or structures being overlooked or not emphasized with the appropriate importance. Clustering methods can inform/automate visualization decisions/processes, ensuring that the attributes chosen to be highlighted best represent the dataset. Automatically-generated colormaps are created based on the clustering results, and these features are highlighted in the render. Thus, physical structures within the dataset are emphasized. Additionally, clusters can identify small features/substructures that may be easily passed over otherwise. It is important that the clusters identified by the clustering algorithms are "physically interpretable" (Milosavljevic et al., 2018), in that each cluster in some 2D phase–space corresponds to a physical structure in the 3D (spatial) dataset. This is useful for scientists to better understand relevant physical processes and enables them to investigate their data in a new way.

Here we outline a new pipeline for cinematic visualization of scientific data, `Estra`, that takes advantage of modern clustering algorithms and applies it to an example simulation of a Moon-forming giant impact (Lock et al., 2018). `Estra` is advantageous to both visualization designers and domain scientists. It is a good starting point for a visualization designer because it reduces time spent on manual data exploration and previsualization. Moreover, it enables domain scientists to explore their data in new ways, and easily create high-fidelity scientific visualizations.

Section 2 describes the post-impact synestia from Lock et al. (2018) used as the example dataset. Section 3 briefly discusses the `Estra` Python workflow, with a full step-by-step process outlined in the accompanying Python notebooks, and gives a theoretical overview of the clustering algorithm used in this work. Section 4 outlines the procedure, from clustering the data and using its results, and describes simple assumptions that we used to build and inform a shader[4] network. Section 5 displays various final renders, and demonstrates the quality and validity of the visualizations. Lastly, we conclude in Section 6. The appendices include a short mathematical treatment of other popular clustering algorithms, as well as additional renders using the `Estra` shader with a perceptually-uniform emissive colormap.

Our code is publicly available at https://github.com/patrickaleo/estra.

2 http://ytini.com.

3 https://unity.com/products/otoy-octanerender.

4 A program that determines how 3D surface properties (lighting, color, etc.) of objects are rendered for each pixel.

2. The Moon-forming terrestrial synestia

As an example dataset to demonstrate our pipeline, we use the output of a smoothed-particle hydrodynamics (SPH) (Gingold and Monaghan, 1977; Price, 2007) simulation of a Moon-forming giant impact from Lock et al. (2018). SPH is a Lagrangian fluid dynamics method where the fluid is divided into particles with the dynamics of each particle governed by interactions with its nearest neighbors. The output is in the form of the properties (spatial position, velocity, thermodynamic properties, gravitational potential etc.) for each particle. Before pre-processing, there are 100,989 particles in our example simulation.

Giant impacts are collisions between planet-sized bodies which are common in the formation of our solar system and exosystems (Raymond et al., 2018), and it is thought that the last impact the Earth experienced ejected sufficient material into orbit to form our Moon (Cameron and Ward, 1976; Hartmann and Davis, 1975). The Moon-forming giant impact was a highly energetic event and left the post-impact body rotating rapidly and with a silicate mantle that was substantially supercritical or vaporized (Lock et al., 2018; Lock and Stewart, 2017; Nakajima and Stevenson, 2015). Lock and Stewart (2017) recently demonstrated that a subset of impacts are sufficiently energetic, and have high enough post-impact angular momentum to produce a previously unrecognized type of planetary structure: a synestia. Synestias could provide a new environment for the formation of the Moon (Lock et al., 2018) and are a topic of ongoing research.

For this work we considered the final time step of an impact that produced a Moon-forming synestia from Lock et al. (2018). In this example, the synestia was formed by a 0.1 M_{Earth} body striking a 0.99 M_{Earth} body spinning with a 2.3 hr period (an angular momentum of three times the present-day Earth–Moon angular momentum) at 15 km s^{-1} and an impact parameter of 0.4. The simulation was run for 48 h of simulation time, when the structure was nearly axisymmetric and had reached a quasi-hydrostatic equilibrium. The output was post-processed to simulate thermal equilibration after the impact, driven by processes that are not captured in the code (see supporting information of Lock et al. (2018) for more details). The outer regions of the synestia were thermally equilibrated and a portion of the outer regions of the body were prescribed to be isentropic. Any condensed material was removed from the simulation and the remaining mass of multiphase particles was forced to lie on the vapor side of the liquid–vapor phase boundary.

The SPH simulation we use has previously been visualized by the Advanced Visualization Lab (AVL) at the National Center for Supercomputing Applications (NCSA) to produce part of an animation for the dome show "Imagine the Moon".[5] Using the same visual effects software Houdini to visualize the dataset, this work directly compares a machine learning clustering-informed cinematic visualization to a custom AVL cinematic visualization.

The synestia dataset in .csv format is available on the `Estra` Github page,[6] for easy replication of our results.

3. `Estra` and clustering algorithms

In this section we briefly discuss the `Estra` Python workflow, with a full step-by-step process outlined in the accompanying Python notebooks, as well as introduce Gaussian Mixture Model (GMM) theory.

3.1. Outline of `Estra` integration with visual effects software Houdini

Once the simulation data is fully loaded into Houdini,[7] we can extract all attribute (parameter) values for each particle in the simulation output as a .csv file using a simple custom script in a Python Script[8] object node. Alternatively, one can extract attribute values from the data file directly. This step is necessary because the clustering algorithms that use the attribute data cannot be performed in Houdini itself. We read in the .csv file into a Jupyter notebook and performed the clustering algorithms utilizing the `sklearn` Python package, and transferred our results back into Houdini to inform some visualization decisions, such as automating a clustering-based color temperature ramp in the material shader. This process is implemented in "Estra.ipynb", and is also detailed in the forthcoming sections. See Naiman et al. (2017) for a simple breakdown of the Houdini graphics user interface (GUI) and a typical workflow session in an astrophysical context, as well as more general background and usage of Houdini.

Houdini accepts data formats which includes, but is not limited to, .geo, .bgeo, .json, .pdb, .obj, and .vdb. Once the simulation dataset is imported from a local directory and into Houdini via a File node in the Network View panel (as referenced via its path-to-file in the "Geometry File" parameter), one can examine all the attribute data – the different parameters included in the simulation proper such as temperature, density, position (x, y, z), etc. – via the "Geometry Spreadsheet" tab. This attribute data will later be used in the clustering algorithms.

3.2. Clustering algorithms

Because this work is crucially dependent on choosing the appropriate clustering algorithm, the Gaussian Mixture Model (GMM)[9] used in this work is explained thoroughly below. Other common clustering algorithms are explained in the Appendix.

A popular and powerful unsupervised learning technique to cluster data is in the form of mixture models—probabilistic models for estimating in which subpopulation within an overall population a datum resides. In GMMs, subpopulations are construed to be Gaussian distributions with unknown parameters, such that all data (the "population") is thought to be generated from a finite mixture of these smaller distributions. Thus, for any one particular data point, there is an inherent probability to which subpopulation, or cluster, it belongs. In other words, a GMM is a parametric probability density function (PDF) for which its components are a sum of weighted Gaussian densities (Reynolds, 2009).

For the algorithm to generate the requisite number of Gaussian mixtures, the user must first assign the number of clusters M, meaning that the number of clusters are known *a priori* or assumed to be known. For GMM clusters to have an ascribed physical meaning, it has to be a reasonable assumption that the dataset can be generated from a superposition of Gaussian distributions.

The GMM mathematical formulation is described below, following the notation of Reynolds (2009). The equation describing the probability of each point (in our case, a particle) being generated by each Gaussian component of the population $p(\mathbf{x}|\lambda)$ can be written as

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i g(\mathbf{x}|\mu_i, \Sigma_i), \tag{1}$$

[5] https://www.adlerplanetarium.org/event/imagine-the-moon/; an excerpt video can be viewed here: https://www.youtube.com/watch?v=7e_6oyROHCU.

[6] https://github.com/patrickaleo/estra.

[7] www.sidefx.com.

[8] https://www.sidefx.com/docs/houdini/nodes/obj/pythonscript.html.

[9] https://scikit-learn.org/stable/modules/mixture.html.

where $\mathbf{x}$ is some D-dimensional data vector particular to the dataset, λ is the collection of variables parameterizing the model

$$\lambda = (w_i, \mu_i, \Sigma_i), \qquad i = 1...M, \tag{2}$$

where $w_{i=1...M}$ are the individual weights of M Gaussian components (constrained to sum to 1), and $g(\mathbf{x}|\mu_i, \Sigma_i)$ for $i = 1...M$ are the various Gaussian component densities. These densities are D-variate Gaussians formulated via

$$g(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right), \tag{3}$$

where μ_i is the mean vector and Σ is the covariance matrix. The covariance matrices can be of several types: 'full', 'tied', 'diagonal', and 'spherical'. In brief, 'full' means full rank covariance, where each component has its own general covariance matrix; 'tied' forces all components to share the same covariance matrix; 'diag' allows for each component to contain their own diagonal covariance matrix; and 'spherical' represents the case where there is a single variance for each component. A 'full' rank covariance was used for this work.

With the type of covariance selected, GMMs will then estimate the various parameters (the mixture weights, means, and covariances, all assumed within λ) using the Expectation–Maximization (EM) algorithm. EM is an iterative algorithm specifically designed to always converge to a local optimum, where parameter values of unobserved latent variables (in this case, the Gaussian components) are estimated by maximizing the likelihood (Dempster et al., 1977). As the name suggests, there is an expectation and a maximization step. After random initialization of the parameters describing the components, the expectation step establishes a function representing the log-likelihood of the data based on those parameters, and by proxy, the latent distribution. In the case of GMM, a probability will be computed for each point being generated by each Gaussian component of the population, $p(\mathbf{x}|\lambda)$. The intermediate goal of EM is to find some new model, λ^*, for which $p(\mathbf{x}|\lambda^*) \geq p(\mathbf{x}|\lambda)$. To achieve this, the maximization step will subsequently tweak the current estimate of the parameters to maximize the log-likelihood established from the expectation step. These same parameters will be used to form the new distribution of the unobserved latent variables (the Gaussian components) in preparation for the next E step. This process continues for a user-specified number of iterations (200 for this work) until some best guess solution is made. Each of the final Gaussian components whose parameters ultimately maximize the log-likelihood, are then defined as the clusters of the dataset.

GMM is best used on flat geometries[10] (obeying Euclidean geometry when measuring distances), and will not inherently bias the cluster sizes to favor particular structures over others. However, it is important to ensure that each mixture has a sufficient number of datapoints (in tandem with choosing a reasonable number of components); otherwise, the covariance matrices become increasingly difficult to estimate, causing spurious estimates of infinite log-likelihoods. A metric to help determine the optimal number of components to describe the data is the Akaike information criterion (AIC) or the Bayesian information criterion (BIC). The particular number of components that produces the lowest AIC or BIC score is potentially the best option to use.

In this work, we chose a 5-cluster GMM with 'full' covariance type, initialized by a random seed.

4. Methods

We now outline the procedure for pre-processing the simulation data, evaluating clustering results, and building a shader within Houdini informed by clustering results.

4.1. Pre-processing

Cinematic visualization can be computationally expensive. Here we impose some thresholds on the simulation data to save computational costs without devaluing the visualization. In this example, the thresholds were chosen based on *a priori* knowledge of simplifying the data in the context of its visualization. This pre-processing is identical to that done by the AVL team in their "Imagine the Moon" dome show visualization, to allow for a side-by-side comparison. Their pre-processing criteria was, in part, motivated by the fact that a large fraction of the simulation volume had slowly-varying material on the outskirts, whose detailed behavior was not critical to understanding the evolution of the synestia's central regions. Thus, processing widely-spaced SPH sample points would have dominated the computation needed for rendering while adding little to the quality of the visualization.

In this work, we threshold two attributes from the simulation: smoothing length (a parameter used to control interactions between particles in SPH (see e.g. Springel et al. (2001)), and density. At very high smoothing length values ($\gtrsim 100$), the sphere sprites[11] of the particles become too large, and subsequently dominate the visualization. However, such particles mostly reside on the outer fringes, and do not constitute the key components of interest. By thresholding to only have particles with $smoothLen < 90$, we saved on computational costs without compromising the visualization.

We also imposed an upper-limit density cutoff of < 3.4 g/cm^3; because the densest particles are mostly within the metallic core of the post-impact body and so add no visual difference to the final render. Only particles that met the aforementioned smoothing length and density thresholds were used in this work, totaling to 35,987 from the original 100,989.

The pre-processed dataset when loaded into Houdini appears as an agglomeration of particles, as shown in Fig. 1.

4.2. Importing clustering results into Houdini

We tested several clustering algorithms in an attempt to find "physically-interpretable" clusters in the pre-processed data; that is, clusters that correspond to significant physical structures within the post-impact body, and not arbitrary, mathematical curiosities. This is important because non-significant or non-real structures emphasized in a final visualization can lead to false conclusions when interpreting the data.

We scaled our attribute values to be of the same order. This is necessary because some machine learning algorithms can be biased towards larger or smaller quantities, as some inherently assume that all attributes are centered about zero and have the same order variance. Thus, if a particular attribute has a variance several orders of magnitude larger than another, it might dominate the objective function and inhibit the estimator. Having performed many clustering tests in different phase–space permutations, we chose a temperature-specific entropy phase space because clusters were easily differentiated and could be physically interpreted. Note that temperature and entropy are conjugate thermodynamic variables and so provide a complete

[10] An overview of clustering algorithms and general usecases can be found at: https://scikit-learn.org/stable/modules/clustering.html

[11] Sprites, or 2D images set in a larger 3D scene, can be attached to particles such that the sprite image always faces the camera (O'Rourke, 1998).

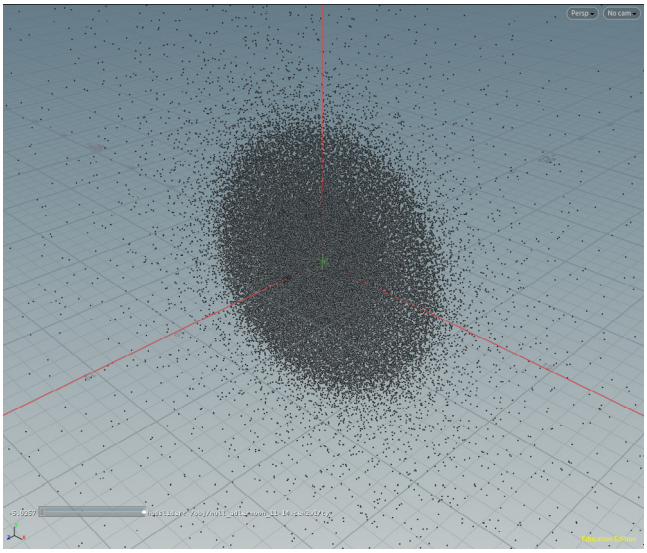

Fig. 1. The Scene View of the synestia, when data is first loaded into the Houdini software. Each individual gray particle is one of the 35,987 SPH particles after pre-processing (Section 4.1).

thermodynamic description making them well suited for describing the thermodynamic phase space. In the units of our example simulation, temperature ($\mathcal{O}(10^3)$ K) is four orders of magnitude smaller than the given specific entropy values ($\mathcal{O}(10^7)$ erg g^{-1} K^{-1}). We scaled these values so that both variables are of the same order using the `StandardScaler` package from the `sklearn.preprocessing` module. Now any of the various clustering algorithms could be used.

Of all the clustering tests we performed, the 5-cluster GMM model was the "best" choice, in that each cluster was found to have a corresponding physically interpretable meaning. Moreover, a 5-cluster model is simple and sufficient to describe the data. What each cluster represents will be discussed in Section 5. The 5-cluster GMM result is shown in Fig. 2.

Examples of poor clustering algorithm choices are shown in Figs. 3 and 4. Fig. 3 shows a 5-cluster Kmeans algorithm in the same temperature–entropy phase space. In this case, data points are split seemingly arbitrarily. For instance, a single cluster of known significance – the outer material prescribed to be isotropic – is split into three clusters. This is likely due to a misapplication of the algorithm—Kmeans is designed to work for a flat geometry (flat manifold) usecase, whereas the data plotted in this temperature–entropy phase space is non-flat.

A second example of a poor choice for clustering was clustering in a density–entropy phase space. As seen in Fig. 4, the distribution of SPH particles in this phase–space globally manifest as a single ribbon-like stream. It is difficult to determine where one cluster (and hence an associated physical structure) ends and another begins with density smoothly decreasing with pressure. Further, this clustering attempt assigns one cluster (red) to what we know to be two different structures: an outer vapor dome region (vertical band) and an isentropic pure-vapor region (horizontal band).

Once a final clustering result was chosen (Fig. 2), the clustering ID results were imported into Houdini to inform the visualization. To do so, we wrote out the `predict` method values to a text file, and imported these values into Houdini via a "Table Import" node. Once these cluster ID values are assigned to an attribute value of the geometry using an "Attribute Create" node, it becomes part of the dataset—all particles in the data are assigned a custom attribute value named "map_id_to_clus", with an integer value ranging from 1–5 representing their cluster ID.

Next, we created an "Attribute VOP" node to read the newly created attribute "map_id_to_clus" as input. With an RGB ramp, we mapped an "map_id_to_clus" value of 1 (cluster ID of 1) to gold $(R, G, B) = (0.9, 0.584, 0)$, the second point to magenta (0.9, 0, 0.911), etc. With this setup, we can display the particles shaded by their cluster assignment in the Scene View, as seen in Fig. 5. This view shows the 3D spatial distribution of the clusters which had been identified in 2D phase space. This step is imperative to determine if the particular clustering algorithm used correctly identifies clusters representative of physical structures within the data. Moreover, the Scene View in Houdini is interactive, such that the user can easily change the viewing angle (see Figs. 5, 6), zoom in/out, and select certain groups of particles to fully understand the data for all time steps (if time-evolving), etc. Outliers in the cluster assignment can also be identified, and their potential impact, if any, on the final render can be determined. For example, three outliers are evident in Fig. 6. It appears that these outliers consist of mixed phase material, and lie well below the liquid–vapor phase boundary in comparison to the rest of the simulation (see Fig. 13). However, the minimal outliers have no visible impact on the final render, so we do not consider them further. It is important to note that we cannot know the ground truth cluster assignments, so the usage of "outlier" simply refers to a different cluster assignment from the overwhelming majority of those around the particle in 3D position space.

Finally, with the clustering results imported into Houdini, we can build a shader and perform the visualization.

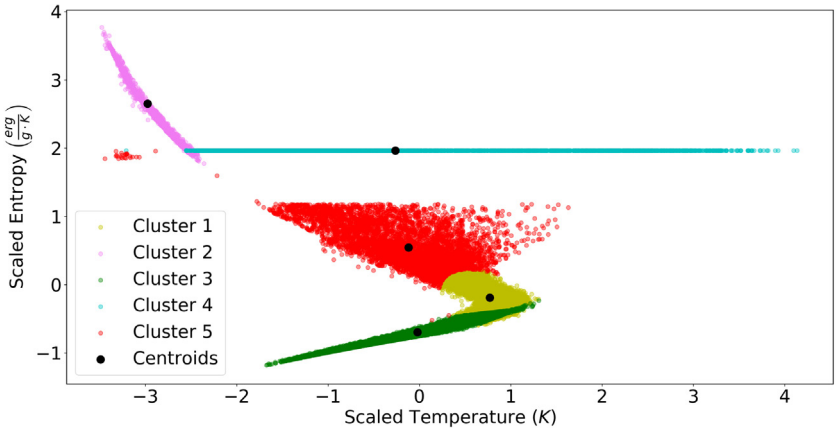

Fig. 2. The results of the 5-cluster GMM algorithm of the 35,987 SPH particles remaining after pre-processing, for a maximum of 200 iterations in a scaled temperature-specific entropy phase space. The covariance type was set to 'full', with weights initialized by 'kmeans' with a random seeding. Each cluster is represented by a different color with the centroids marked by black circles.

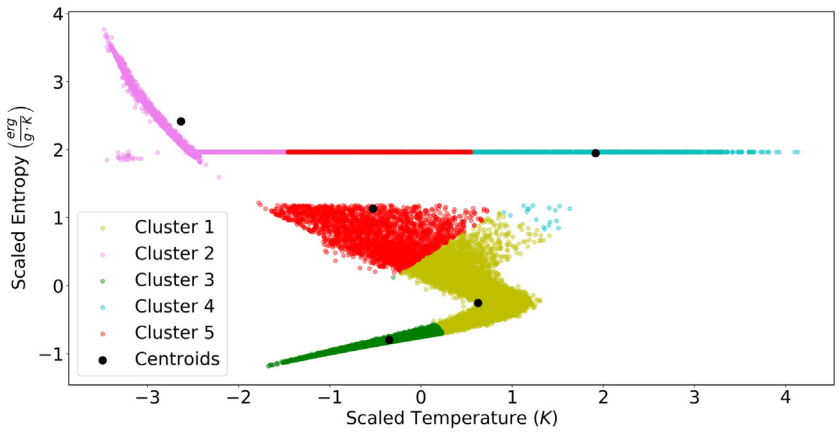

Fig. 3. The results of a 5-cluster Kmeans algorithm, for a maximum of 300 iterations in a scaled temperature–entropy phase space. The weights were initialized by 'k-means++' in `sklearn.cluster.KMeans`, with 50 runs set by different centroid seeding. Each cluster is represented by a different color with cluster centroids marked by black circles. After pre-processing, there are 35,987 SPH particles to be clustered, ran with the same random seed as in the GMM clustering algorithm, the results of which are shown in Fig. 2. This is an example of a poor clustering algorithm choice in this phase–space because it arbitrarily splits likely physical clusters (such as a singular isentropic structure) into multiple disparate clusters. Further, Kmeans is designed to be used on flat geometry, but in this phase–space the data is non-flat.

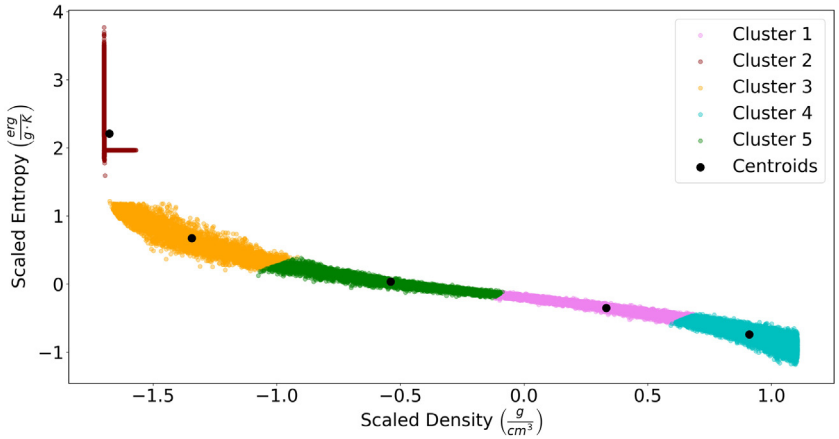

Fig. 4. As Fig. 3, but with the Kmeans clustering performed in a scaled density–entropy phase–space instead of a scaled temperature–entropy phase–space. This is also an example of a poor phase–space choice because most of the data is in one streamlined band, and it is difficult to determine where one cluster begins and another ends. Also, one cluster (red) incorrectly groups together two known significant structures: a vapor dome region (red vertical band), and an isentropic pure-vapor region (red horizontal band).

4.3. Algorithmically generating colormaps from clustering results

An advantage of clustering the data is that it allows for the automatic creation of transfer functions or color ramps which are informed by physical structures in the data. Houdini uses the transfer function (a set of (R, G, B, α) values for a range of data values for a variable, which here is temperature) in shading the final image.[12] In building a color ramp, the shader requires 'position' markers called key points with associated (R, G, B) values. For instance, although a file with color map data may have 1024 rows of (R, G, B, α) values for an emissive blackbody color scheme, Houdini requires only a handful of position markers containing (R, G, B) values, and uses a user-defined interpolation function (e.g. Linear, B-Spline, Bezier, Catmull) to create the ramp. Then, once a range of temperatures are defined for that color ramp, each temperature in that range will be assigned a unique (R, G, B) value in the shader.

For this work, each position marker with its unique (R, G, B) value is informed by the clustering results. First, the entire temperature range is rescaled back to physical units (Kelvin) using the `inverse_transform` method in `sklearn`. The range of temperatures is then normalized to be between 0–1 (as position markers in Houdini have values [0,1]). The minimum, mean (represented by the centroid), and maximum temperature values across all members of each cluster are assigned a 'position' marker value from [0,1] based on where they lie on the normalized scale. Lastly, with each key points' position value determined, its corresponding (R, G, B, α) value in the imported color scheme is set.[13] In other words, position markers from each cluster correspond to Houdini key points on the color ramp.

We used a modified emissive color scheme from the AVL's catalog for our final renders. It maps any temperature value to

[12] While the standard definition of a "transfer function" deals with (R, G, B, α), Houdini splits this into a "color ramp" of (R, G, B) values and a "spline ramp" for α.

[13] Where in this work a temperature range [0, 14308 K] was normalized to [0,1], a temperature of 6000 K would have a marker position value of 0.419. This would correspond to the 429th row of the 1024 row color map.

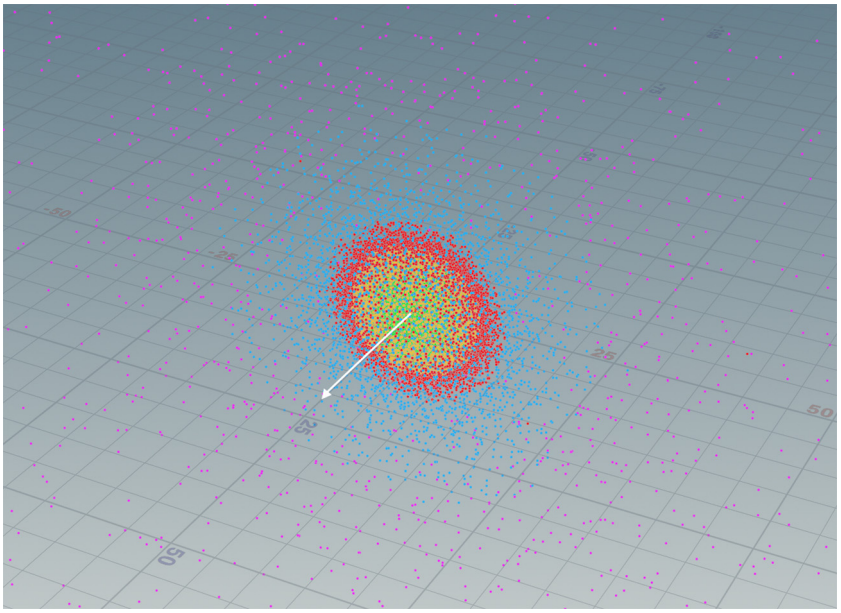

Fig. 5. Scene View of the synestia dataset, colored by its clustering ID results from the 5-cluster GMM result of Fig. 2. Thus, the color corresponding to each particle in 2D phase space can be seen in 3D position space in the Houdini software. The white arrow marks the rotation axis.

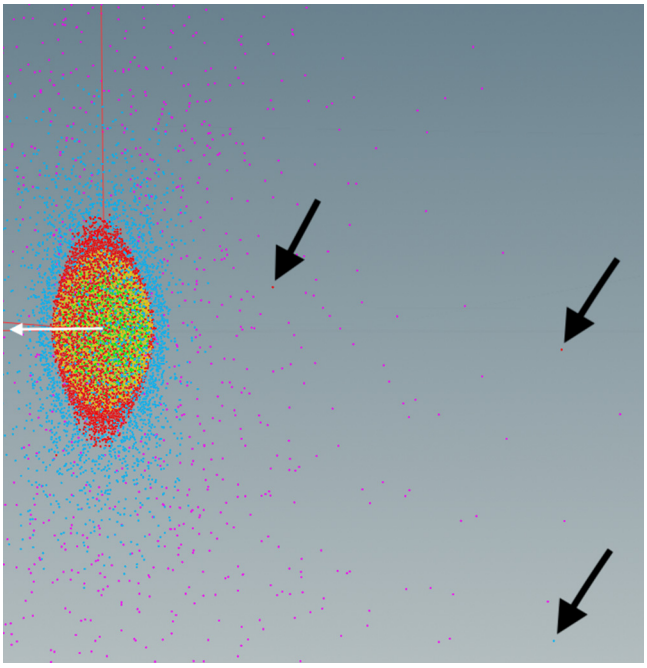

Fig. 6. A zoomed-in, rotated view from Fig. 5, where the rotation axis is along the horizontal extending out from the synestia bulge, as shown by a white arrow. On this scale, we can see individual particles and their cluster association. Individual outliers are seen, whereby an outlier is defined as being a particle with a different association relative to surrounding particles in physical space, marked by the large black arrows. These outliers are from the left-most grouping of red and cyan particles at $(-3, 2)$ in Fig. 2, where these few particles are possibly assigned an incorrect cluster. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

its associated (R, G, B, α) color value using an emissive blackbody color temperature scale, and is non-perceptually uniform[14].

Our final clustering-informed temperature ramp is in Fig. 7(a), and the AVL's custom temperature ramp (used in the "Imagine the Moon" dome show) is in Fig. 7(b).

4.4. Building a custom shader

To create the visualization, Houdini requires a shader to tell its native renderer, Mantra, how light interacts with each particle in the scene.[15] Once established, the scene can be rendered, where the three-dimensional scene is transformed into a two-dimensional image. Although there are built-in shaders, none of them are tailored to emissive, SPH data. In the "From_Results_to_Render.ipynb" notebook, we detailed how we created a general-use particle sprite shader for emissive material, with values tailored to our example dataset. Future projects can simply copy this shader network into their own Houdini scene files, and modify the values appropriately. We summarize the process of building our shader below.

Because the material in SPH simulations is, as the name implies, divided into particles, it is best to design the visualization using particles. Thus, we transformed each data particle into a sphere sprite with a non-uniform rational B-spline (NURBS) primitive type.[16] To do so, we created a sphere primitive inside a geometry object,[17] and set the sphere's "Primitive Type" to NURBS to maintain smooth surface continuity. Later, in an Instance node used for our final render, we linked its "Instance Object" parameter to the geometry object in which the sphere was embedded, such that each SPH particle was rendered as a sphere. Also, because each simulated SPH particle has a unique size or radius, we forced its sphere sprite to retain its smoothing length value from the data by assigning it to each particles'

[14] If desired, there are other perceptually uniform color maps available to use from e.g. Moreland (2016) and https://github.com/liamedeiros/ehtplot. Some additional perceptually uniform renders of this dataset are found in the Appendix.

[15] Many other popular renderers such as Renderman will work, but may involve different steps to create a shader.

[16] https://www.sidefx.com/docs/houdini/model/primitives.html#nurbs.

[17] https://www.sidefx.com/docs/houdini/nodes/sop/sphere.html.

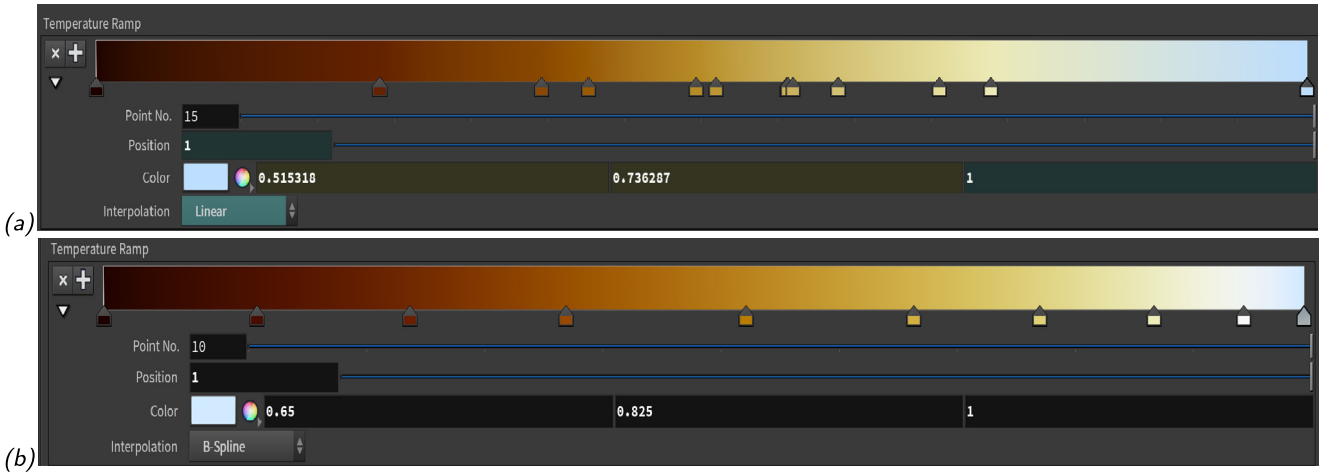

Fig. 7. *(a)*: The non-uniform, emissive colormap with markers and associated (R, G, B) values informed by the 5-cluster GMM result. The colormap is over temperatures [0 K, 14308 K] to match that of the AVL rendering for later comparison, despite the dataset temperatures ranging from [2453 K, 14308 K] (the range [0 K, 14308 K] was chosen for aesthetic reasons). Here, the hottest temperatures map to light blue, moderate temperatures map to orange, and the coolest temperatures map to dark red/brown. *(b)*: The custom AVL colormap, from [0 K, 14308 K], used in the "Imagine the Moon" dome show. Ten key points are used with a B-Spline interpolation, and are roughly equally spaced over the length of the colormap. Here the hottest temperatures map to mostly light yellow/white tones instead of a light blue like in the `Estra`-generated colormap. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Houdini "pscale" value[18]. Lastly we duplicated our pre-processing step as described in Section 4 to ensure only the data retained in the clustering steps (and aligned in the same order) is used in the final visualization.

To build a custom shader, a Material Shader Builder node[19] was modified. Parameter nodes[20] and ramps[21] were created to read in the data's temperature, density, and smoothing length values, because these control the emission color and opacity over a range of values. For example, a spline ramp type was used to adjust the transparency of each particle based on its smoothing length value. In this simulation, the largest particles (largest pscale) reside on the outer fringes of the simulation, as pscale is related to how many and how close its nearest neighbors are; the smallest particles/lowest pscale values constitute the central region of the post-impact body. Mapping large pscale values to low opacity/high transparency and small pscale values to high opacity/low transparency allows for the whole simulation to be seen and visualized; otherwise, the largest particles would dominate the visualization, and obscure the rest of the data.

In choosing the mapping transfer function for the temperature, density, and smoothing length opacity ramps, we made simple assumptions to enable ease of use. We assumed the density (opacity) ramp to have the form of a cubic Bezier curve, governed by the parametric equation:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3, 0 \leq t \leq 1, \quad (4)$$

with control points $\mathbf{P}_0 = (0, 0)$, $\mathbf{P}_1 = (0, 0.2)$, $\mathbf{P}_2 = (0, 1)$, and $\mathbf{P}_3 = (1, 1)$, where t is time (to trace out the full curve, not to be confused with a physical time associated with our dataset). These control points define a control polygon from which our curve is drawn. This resulting curve represents values for the position and value markers on the density ramp. We chose five values to define the ramp traced out by the curve: (0, 0), (0.1, 0.53), (0.25, 0.75), (0.5, 0.91), (1, (1), as shown in Fig. 8.[22] These specific control points were chosen because: (1) the rectangular box of the ramps have domains $x \in [0, 1]$, $y \in [0, 1]$; (2) this choice results in a good balance of opacity as a function of density, whereby particles of all densities are visible, and are not obscured due to the extreme values—the densest particles (in the central region of the body) are visible without drowning out the least dense particles (on the edges) and vice versa.

When plotting a histogram of the number of particle counts as a function of binned density, there is a double peak at the lowest and largest density values (Fig. 9). A cubic Bezier curve is thus a good starting point to allow a particle to be simultaneously more opaque and emissive with increasing density (as this density ramp is connected to both opacity and emission multipliers), while allowing for the many contained lowest density particles in the center to be visible and not overpower the visualization. This cubic Bezier curve will be useful in many applications where dense objects are embedded in less-dense mediums, e.g. star formation.

The temperature ramp controls the color as a function of temperature. Its values are the result of the GMM clustering algorithm, which is the transfer function shown in Fig. 7(a).

Lastly, the pscale ramp also controls opacity. Its values are the transposed values of the cubic Bezier curve used in the density ramp. This way, the largest pscale values (largest key position marker values on the ramp) have low opacity/high transparency (low ramp y-values), and vice versa.

Once the temperature and pscale ramps were set, a test image was rendered, shown in Fig. 10(a). This render has a hot inner region and a cooler outer region, but the look is clumpy and the center of the post-impact body is dim. With many of the interesting features occluded, more work was done.

Because luminance from each sphere sprite is isotropic and its intensity follows Lambert's cosine law, the screen–space distance

[18] While "smoothing length" is a data variable, "pscale" refers to a multiplier in Houdini on the size of a particle. We assigned the value of smoothing length to a particle's pscale. Conceptually these terms are different, but as far as implementation in Houdini is concerned, they are the same. We used the smoothing length to drive the size of the particle, by assigning it to pscale, or "particle size multiplication factor".

[19] https://www.sidefx.com/docs/houdini/nodes/vop/materialbuilder.html.

[20] https://www.sidefx.com/docs/houdini/nodes/vop/parameter.html.

[21] https://www.sidefx.com/docs/houdini/nodes/vop/rampparm.html.

[22] The ramp starts at an arbitrary non-zero y-value (0,0.03) so that the particle(s) with the lowest densities still contain some opacity, and are not completely transparent in the render.

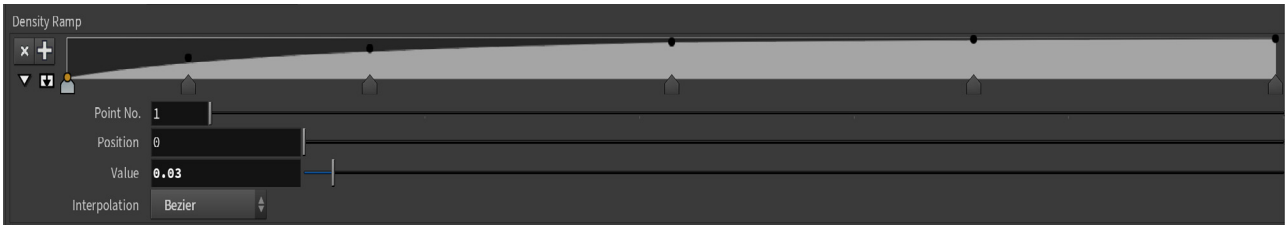

Fig. 8. The density ramp, mapped to opacity in the shader, where density values increase from left to right from 0 to 3.4—its max value after pre-processing. The higher the 'Value' indicated by the ramp, the more opaque (less transparent) each particle at that particular density will be. Values are determined by a cubic Bezier curve with the control points $\mathbf{P}_0 = (0, 0)$, $\mathbf{P}_1 = (0, 0.2)$, $\mathbf{P}_2 = (0, 1)$, and $\mathbf{P}_3 = (1, 1)$. The 'Interpolation' between each point is also set to 'Bezier'. Less dense particles (typically near the outer edges of the synestia) are less opaque, and more dense particles (typically concentrated in the center of the synestia) are more opaque.

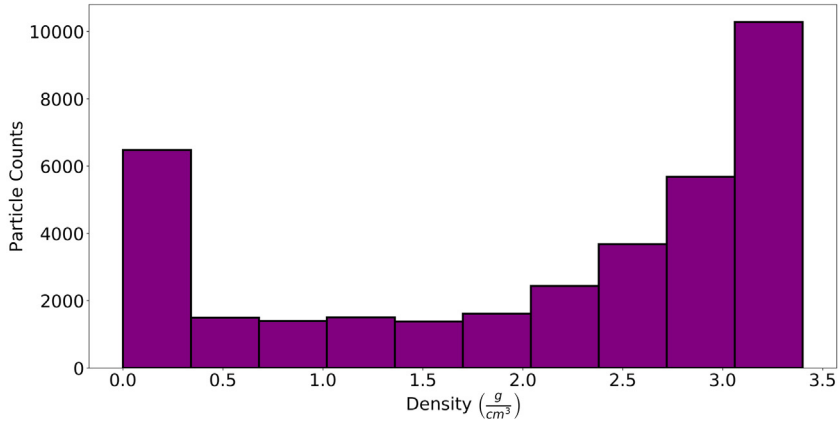

Fig. 9. A histogram of the number of particle counts as a function of binned (pre-processed) density. Most particles are either in the highest density regimes (center of post-impact body) or the lowest density regimes (out in the outer fringes). For our purposes, the higher density regions of the simulation are most interesting to visualize, in part to better compare this work's renders to those from the AVL.

from the viewing center was calculated to inform the intensity falloff. This falloff is dominated by the Gaussian function and is known as a Gaussian falloff profile. The screen–space distance d_{screen} is $\sqrt{1-(\mathbf{N}\cdot\mathbf{I})^2}$, where $\mathbf{N}$ is the (normalized) normal vector from the sprite surface, and $\mathbf{I}$ is (normalized) vector for the incident light. In this manner, the screen–space distance is largest when $\mathbf{N}$ and $\mathbf{I}$ are orthogonal, i.e. in the line-of-sight of the viewing angle, and is smallest when $\mathbf{N}$ and $\mathbf{I}$ are unidirectional. A node network was made to perform the screen–space distance from viewing center calculation. Once completed, its output d_{screen} value was used as an input in another node network to calculate a Gaussian falloff profile for each sphere sprite. This Gaussian falloff node network use the standard Gaussian function $g(x)$ of the form

$$g(x) = ae^{\frac{-(b-x)^2}{2c^2}}, \tag{5}$$

where the parameter a is the height of the curve's peak, b is the position of the center of the peak (the d_{screen} value) and c (the standard deviation) controls the width of the "bell". A Gaussian profile is a good approximation for the spectral intensity profile, because each sphere is a resolved point source. The point spread function (PSF) that represents the light distribution of such a point source is approximately Gaussian (Sterken and Manfroid, 1992).

With these new node networks added to our shader, another test render was created, and is shown in Fig. 10(b).

Finally, the clouds of gas and dust are randomized with noise to visually suggest sub-grid scale fluctuations, as a result of turbulent motion. These noise values, as well as other parameters such as the Gaussian profile constants, can be fine-tuned to achieve the purpose of the visualization.

With final noise values and Gaussian profile constants decided, our shader is completed and a final render is made, as shown in Fig. 10(c).

5. Results & discussion

5.1. Comparing Estra and AVL renders

A zoomed-in view of Fig. 10(c), the final `Estra` render using GMM clustering which focuses solely on the center of the synestia structure, is shown in Fig. 11(a). For comparison, the manually-designed AVL render, which utilizes a more complicated shader and is used in the full dome show "Imagine the Moon", appears in Fig. 11(b) with the same scale and aspect ratio. The `Estra` rendering is qualitatively similar to the AVL rendering. Both retain the same bright bulge, and have a dusty ring of material on a plane perpendicular to the rotation axis. Additionally, both are emissive, and have the same clumping of gas and dust, with Gaussian falloff, and a similar color palette. The AVL rendering has more red tones, and less bright white highlights of the hottest material in the bulge, due to the slight difference in color maps. Although the color map used for The `Estra` rendering (Fig. 11(a)) is a simplified and extended version of The AVL rendering (Fig. 11(b)), it contains a different transfer function based on the clustering results. For example, moving the position markers (which themselves are directly determined from the minimum, mean, and maximum values of each cluster—see Section 4.3), changes how temperature is mapped to a particular color. The spacing width between the position markers on the ramp controls the rate at which the color changes; the larger the spacing between the markers, the greater the change in color space and vice versa. Hence, dissimilar phase–space clusters have a wider color palette spread for easier visual identification in the `Estra` render.

It is important to keep in mind that both renders cannot produce a fully realistic image; there is not one particular "right" or "correct" render. In fact, the purpose of this work is to enable

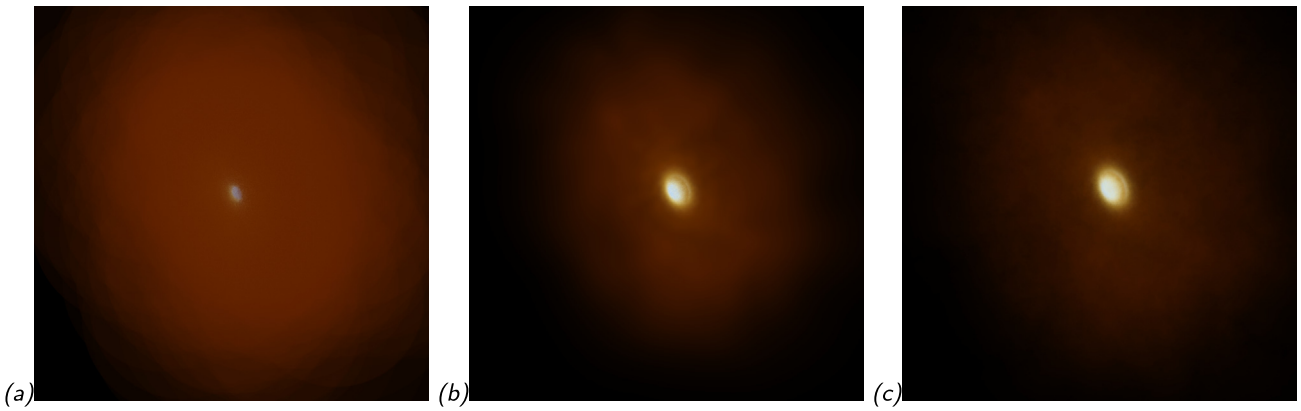

Fig. 10. *(a)*: This test render uses an incomplete shader, after temperature and pscale ramps are created. Visually it is evident from the color mapping that we have a redder, cooler outer region and a hotter, whiter/bluer inner region. *(b)*: This test render, of the same scale and aspect ratio as *(a)*, also uses an incomplete shader, but is after a screen space distance d_{screen} is calculated and used in a Gaussian falloff profile for each sphere sprite. Because there is little clumping of the dust and gas clouds, randomized noise needs to be added to visually suggest sub-grid scale fluctuations. *(c)*: The final render, after random noise is added to complete the `Estra` shader. Because this view is distant from our areas of interest, a close-up of this same render is found in Fig. 11(a). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

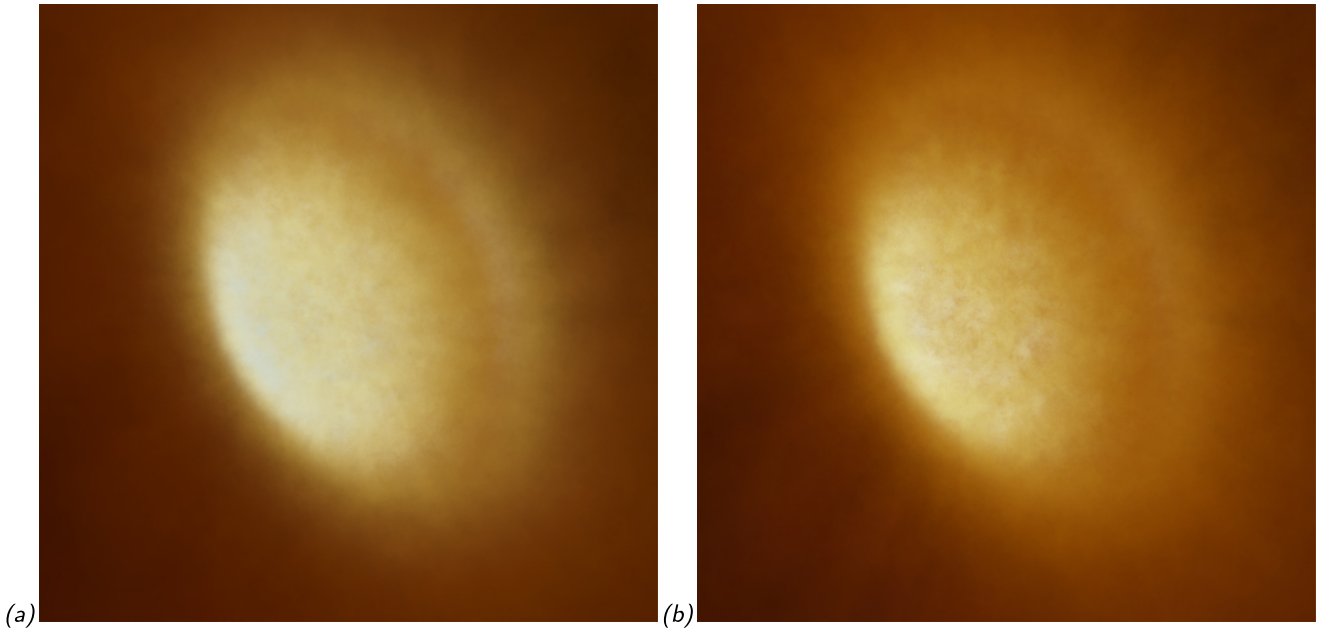

Fig. 11. The final renders of the synestia (of the exact view shown in Fig. 1) using *(a)* the `Estra` shader—with its colormap informed by the 5-cluster GMM result (Fig. 7(a))—and *(b)* the AVL's more complicated, custom shader not informed by a machine learning algorithm (colormap from Fig. 7(b)). Although choosing the appropriate clustering algorithm and developing the pipeline took one of us (P. D. Aleo) approximately two months, (*a*) was created in approximately a day's work once the workflow was established. Meanwhile, the timeframe for (*b*) was similar, but involved the work of three visualization designers (A.J. Christensen, K. Borkiewicz, R. Patterson). This demonstrates how a scientist can create a simplified visualization of their own work, comparable to the quality of one produced by a professional visualization team.

a scientist with little to no background in cinematic visualization to visualize their own work, and we have shown that with clustering-informed color-mapping and simple assumptions, one can create compelling visualizations aesthetically similar to ones produced by visualization teams for full dome productions.

The `Estra` pipeline is also potentially useful in the production pipelines of visualization teams. They can create clustering-informed visualizations to reduce time spent on data exploration, and simply tweak or complicate them as necessary to match their needs.

Once the shader network is setup and pre-processing is completed, it is trivial to try any number of clustering algorithms and import the cluster IDs into Houdini. Building the general purpose shader network from scratch takes approximately an hour, and running a clustering algorithm and importing clustering results takes half that. The only significant time bottleneck is the quality of the render: a 4096 × 4096 render like Fig. 11 took ∼8 h on an HP z820 workstation with a single Xeon E5-2670, 2.5 GHz 10-core processor. If the number of particles used was reduced, or the pscale threshold and/or image quality was reduced, this render would be less computationally expensive.

5.2. Finding physically interpretable clusters in the synestia

One of the key aims of this work is to find and visualize "physically interpretable" clusters, whereby each cluster has a corresponding physical meaning or structure to automate and guide the visualization. With the color-mapping process designed to highlight individual clusters, physical interpretability is a requirement for the resulting visualizations to be meaningful, and not

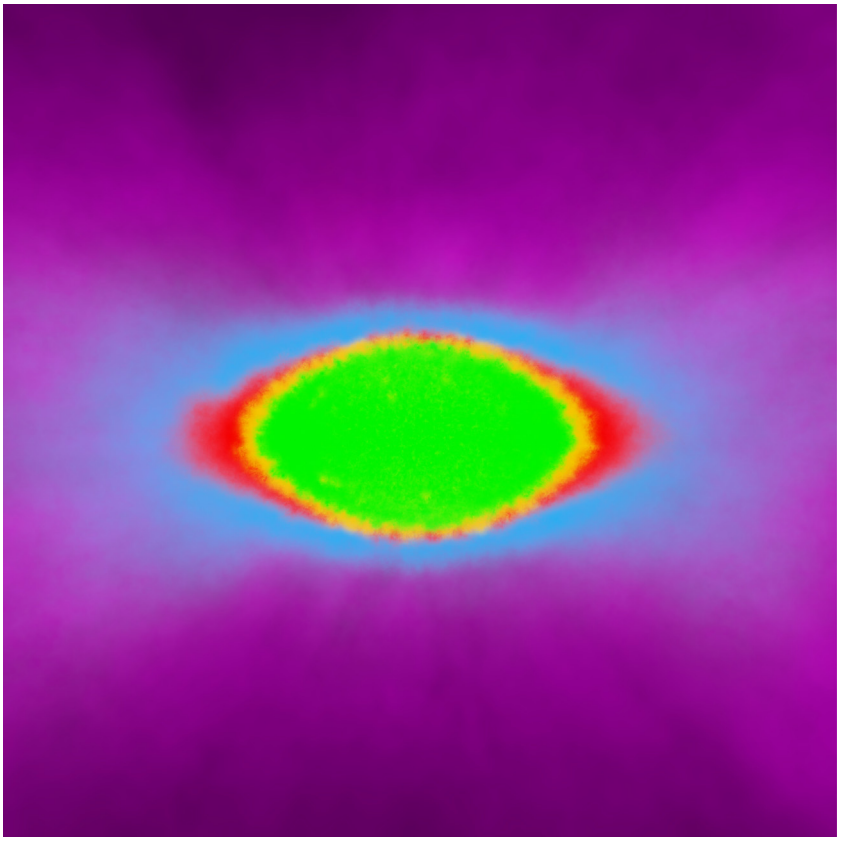

Fig. 12. A false-colormap render a cut-through of the far half of the synestia to emphasize the physical manifestation of the clustering results. The colors shown here match to those of temperature–entropy phase space plot (Fig. 2) for easy comparison. The green region is the "lower mantle" of the synestia; yellow is the "transition region" containing a rapid increase in entropy with depth; red is the "supercritical region" of highly shocked silicate material which typically has specific entropies greater than the critical point entropy and so is either supercritical fluid or high-pressure vapor; cyan is the "isentropic pure-vapor region", because its constituent particles were forced to be isentropic during post-processing; and magenta is the outer "vapor dome region", where particles lie along the vapor side of the liquid–vapor phase boundary. This is rendered with the same custom shader as in Fig. 11(a), with the only difference being the temperature ramp, which here maps by cluster ID and not by assigning color to a temperature transfer function from GMM results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

arbitrarily representations of the data which cannot be soundly analyzed.

In order to understand which clusters are physically meaningful, our 2D phase space cluster results were imported into Houdini's Scene View. In `Estra`, every particle is assigned a cluster membership, and this cluster membership ID becomes a new data attribute. Each particle retains its unique cluster assignment, and from there the shader can be extended to assign a unique color to each cluster. To do so, one can either create a new color ramp or copy an existing one, set its interpolation to 'Constant', and map discernible colors to match the number of clusters. Once the clustering assignment ID is mapped to a color, each particle will be colored by its appropriate grouping. With Houdini's node network, all one has to do is change the connection to any one of the color ramps to achieve the desired render.

In this example, each cluster color pairing (Cluster 1–Gold, Cluster 2–Magenta, etc.) as shown in Fig. 2 is retained in the shader. Each particle's 2D phase space cluster assignment manifests in 3D position space, from which the wide array of Houdini tools can be used for data exploration—rotating about the data, taking slices, isolating one cluster, etc. Seeing and interacting with the cluster-colored data in 3D position space, combined with the knowledge of the simulated data, enables the researcher to determine if the resultant clustering assignments have physical meaning, or "interpretability". If the result appears nonsensical or arbitrary, likely another clustering algorithm should be used and further explored.

For our example post-impact synestia, the 5-cluster GMM result is shown in Fig. 12 (with earlier examples already displayed in Figs. 5, 6). It is rendered in Houdini with a false colormap identical to the color scheme used in Fig. 2. This render is oriented such that the rotation axis points upwards through the central region. Also, to clearly see the physical meaning of the clusters, this render is a cut-through of the far half of the dataset, such that particles on the near side of the camera (our current view) do not occlude those on the far side along the line of sight through the synestia. In other words, only the far-half of the synestia is rendered, enabling us to investigate its inner layers via our line of sight.

A motivating factor for choosing the 5-cluster GMM result is that it best clusters the data into its constituent components: the green region is the "lower mantle" of the synestia which experiences only moderate heating during the impact and so is still of relatively low entropy; red is the "supercritical region" of highly shocked silicate material which typically has specific entropies greater than the critical point entropy and so is either supercritical fluid or high-pressure vapor; yellow is the "transition region" between the supercritical fluid (red) and lower mantle regions (green); cyan is the "isentropic pure-vapor region", because its constituent particles were forced to be isentropic during post-processing (see Lock et al. (2018)); and magenta is the region where particles lie along the vapor side of the liquid–vapor phase boundary in the outer "vapor dome" region. For the first time, the different components are easily discernible and can be understood in their proper context. Lastly, it is important to remember

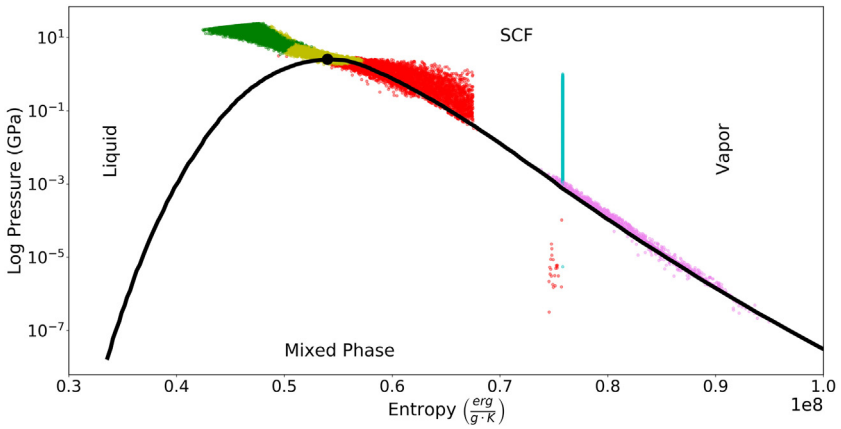

Fig. 13. Our 5-cluster GMM results plotted in a specific entropy–pressure phase space, where cluster membership is the same as in Fig. 2. In this phase–space, the liquid–vapor phase boundary is a dome-shaped curve (black line). The black dot on the vapor dome is the critical point for the equation of state used in these simulations (S_{crit} = 5.40e7 erg K^{-1} g^{-1}, p_{crit} = 2.55 GPa, T_{crit} = 8,810 K, ρ_{crit} = 1.68 g cm^{-3}). Material to the left of the dome is liquid, material to the right of the dome and below the critical point is vapor, material above and to the right of the critical point is supercritical fluid (SCF), and material within the dome is a mixture of both liquid and vapor. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

that these associations are interpretations by those who created the simulation.

As an additional check to confirm our structure analysis, we plotted the 5-cluster GMM results against the liquid–vapor phase boundary in pressure-specific entropy space, shown in Fig. 13. The yellow transition region between the green lower mantle and the red supercritical region contains the critical point. The critical point marks the transition between liquid and supercritical fluid/liquid, a distinct change in the thermodynamic properties of the silicate, and so has been correctly identified by the clustering algorithm as a key region connecting, yet distinct from, the inner liquid and supercritical regions. Our clustering interpretation has also identified the region that constitutes supercritical fluid or high-pressure vapor as a distinct group, and the magenta outer vapor dome region neatly incorporates the particles that lie on the vapor side of the liquid–vapor phase boundary. Additionally, the cyan isentropic pure-vapor region had been clearly separated from the rest of the structure and is distinct from the magenta vapor dome cluster, thus confirming that the Kmeans clustering algorithm, which combined the pure-vapor and isentropic clusters (see Fig. 4), would indeed have been a poor choice. Our 5-cluster GMM algorithm distinctly separates physically significant regions within the synestia.

Retaining the same view as Fig. 12, we can re-render a scene with both the non-uniform, emissive shader as in Fig. 11(a) and the same temperature ramp as in Fig. 7(a), shown in Fig. 14. Here, we can nearly discern the five structures as they appear in their individually colored cluster assignment. The innermost lower mantle is slightly cooler (more brown) than the transition region and supercritical region. This makes sense, as the supercritical region contains more highly shocked material than that in the lower mantle, and is of a higher temperature (yellow/white). The isentropic region contains the hottest material (most blue) on the outer surface layer perpendicular to the rotation axis material, which gets colder as the material occupies the outer edges of the wings. This is evident from the isentropic feature in Fig. 2. However, because of this temperature change, it is hard to tell to what extent this isentropic feature is due to this emissive shader, unless one looks at Fig. 14 for reference.

It is important to note that the stark change in temperature from the moderate, light yellow transition region to the hot, blue isentropic region is an artificially enforced boundary due to the post-processing of data by the scientists who created the simulation. This post-processing is designed to simulate thermal equilibrium of the post-impact body by processes not captured

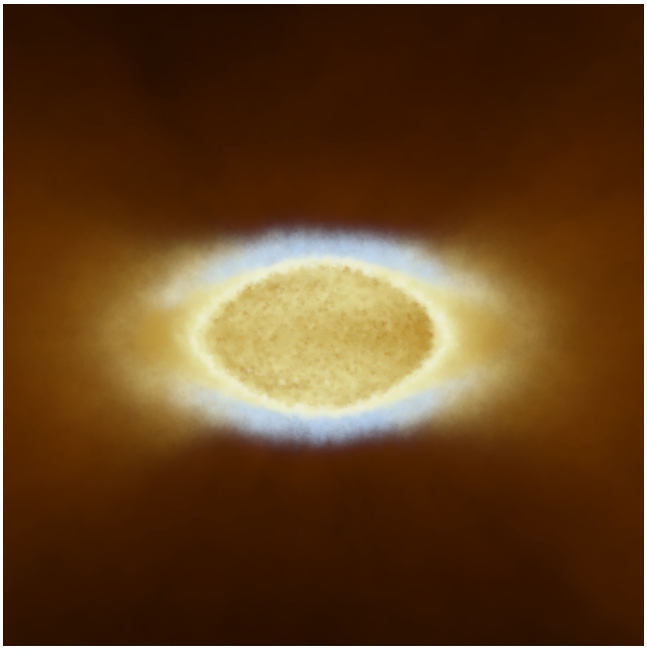

Fig. 14. Same as Fig. 12, but rendered with both the non-uniform, emissive shader as in Fig. 11(a) and the same temperature ramp as in Fig. 7(a). We can clearly see some individual physical components of the synestia. From this, the hottest component of the synestia (light blue) – the isentropic region – is the outer surface layer perpendicular to the rotation axis, and not the innermost central region. Similar views can help scientists understand how their data is structured, and how different parameters map to those structures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in the original SPH code but has the unfortunate consequence of introducing an unrealistically sharp boundary in the structure. In actuality the transition between these layers will be more gradiated. An advantage to a scientist visualizing their own work is that they will know artificialities such as this and can take steps to reduce their prominence in visualizations.

The coolest material (dark brown) – which also has the highest specific entropy of silicate material – lies along the vapor-phase boundary in the outer edges of the bulge and disk. This, too, makes physical sense, as although the outer layers contain highly shocked, high-entropy material they are at low pressures where

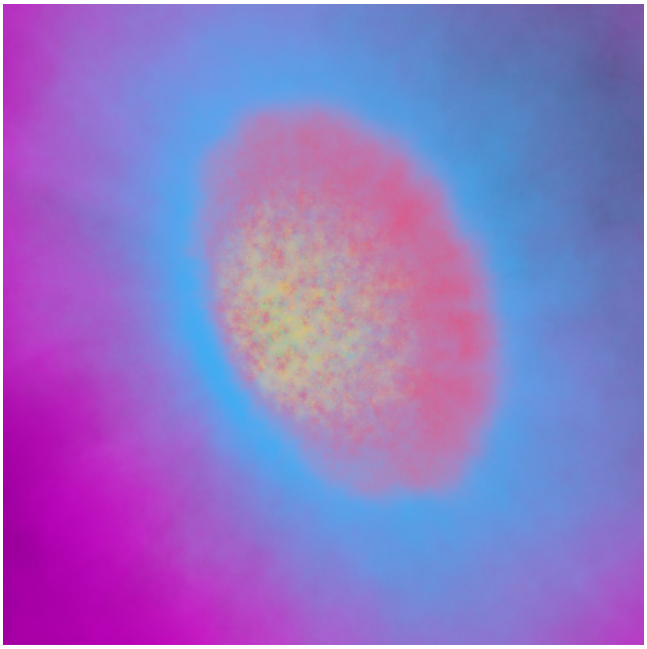

Fig. 15. Same as Fig. 11(a), except the colormap is discrete and colored by the same 5-cluster GMM assignment as in Figs. 2,12.

the temperature on the liquid–vapor phase boundary is relatively low.

Similarly, we can render the same view as in Fig. 11(a) but shading with the cluster assignment colors from the GMM, seen in Fig. 15. From this angle (no slicing involved), it appears that the dusty ring of material is dominated by material from the supercritical region cluster. The brightest bulge material originates from the hottest isentropic particles. This is most apparent in Fig. 11(a) as opposed to Fig. 11(b), as intended. The spotty/patchy appearance of different temperature material in the bulge center in this line of sight is due to our view containing contributions from several different layers of the synestia.

5.3. Validating the visualizations

Validating cinematic visualizations is a long-standing problem in the field, in essence due to its inherent nexus of art (visual effects) and science. Particularly, it is challenging to accurately represent the data in a way that is understandable to experts and non-experts alike (Borkiewicz et al., 2017). Even in the case of this work, where the purpose of the visualization is tailored towards experts, how does one quantify a validation metric to ascribe its "goodness"? When is the visualization "correct" enough to where it can be published in an academic journal or consumed by a general audience? With a medium that mixes qualitative and quantitative regimes, there is still no singular answer, but here we discuss several conventions.

Whilst creating the visualization, visualization teams consult peers and domain scientists – especially those responsible for producing the data – frequently for feedback. This includes, but is not limited to, explaining the purpose of the simulation and establishing its context in the field at large; checking factual accuracy on a documentary script; rerunning, fixing, or filling in gaps of the simulation; and, most importantly, vetting that the visualization does not convey essential aspects of the simulation that is objectively wrong (Borkiewicz et al., 2017, 2019a). It is up to the experts involved in both the visualization and science communities to agree that the visualization is an accurate (or accurate as possible) representation of the data. However, what "accurate" is in this context depends on the nature or purpose of the work. Visualizations are not a complete and holistic depiction of reality; some aspects are presented at the expense of others. Thus, the criterion for validity is not an absolute accuracy, but whether the visualization communicates some points effectively, or clarifies and reveals relationships between variables or physical processes, etc. Subsequently, this implies that accuracy or validity also depends on the purpose of the work.

Despite not being a concretely defined metric, accuracy is an important requirement because it is impossible to guess in the future who will see the visualization and in what context, and subsequently be misinformed. In fact, people psychologically tend to believe visualizations are true (Borkiewicz et al., 2017, 2019a), and thus if the final visualization conveys a biased or incorrect message – whether intentionally or unintentionally, even if based on unbiased data – audiences can fall prey to scientific misconceptions.[23]

When the cinematic visualization is designed for consumption by general audiences, determining its success is harder. Audience testing involves entrance and exit interviews or questionnaires, and are meant to document audience reactions as opposed to learned intuition (Borkiewicz et al., 2017, 2019a). As expected, assessing the visualizations' success is not easily quantifiable. Further complications arise when animations of highly specialized dynamic subject matter are visually complex, which can negatively affect the learning experience (Lowe, 2003).

A third metric is an "eye test", where different representations/colors/camera positions, etc. of the same dataset are shown to different audiences, but all of whom are given the same questionnaire. Although this cannot establish the baseline of how "good" a visualization is, it can inform which visualization is "better" than another in whatever its particular usecase.

In the case of this work, the cinematic synestia visualizations are for field experts. Further, our clustering-informed visualizations have a tractable interpretation of its phase–space clusters, put into proper context of the physical structures and processes at play, thus completing the goal set by this work. This is as close to a metric of "accuracy" we can achieve. Future studies would benefit from some quantitative metric by which one could ascribe a "goodness rating" to the final visualizations, but that is beyond the scope of this work.

6. Conclusion

We have demonstrated the feasibility of using the visual effects software Houdini for cinematic astrophysical data visualization, informed by machine learning clustering algorithms. We outline a step-by-step process from a raw simulation into a finished render that can be utilized by non-experts in the field of visualization to achieve production-quality outputs. We used machine learning clustering algorithms and simple assumptions to inform the visualization process via our Python pipeline `Estra`. As proof of concept, we used a single timestep of a post-impact, thermally-equilibrated Moon-forming synestia from (Lock et al., 2018).

We showed the results of a 5-cluster GMM algorithm, which clustered the data into five distinct, physically-meaningful or "interpretable" clusters: a lower mantle region, transition region, supercritical region, isentropic pure-vapor region, and outer vapor dome region. By having the minimum, mean, and maximum values of these clusters in temperature–entropy phase space inform the colormap, we are able to emphasize these distinct

[23] For a holistic overview of how and why people become misinformed about scientific concepts, see Scheufele and Krause (2019) & references therein.

structures in the render. Moreover, by assigning each cluster membership as an attribute and read into Houdini, each 2D phase–space cluster can be displayed in 3D position space. This will enable any researcher to better understand their data and better interpret the clustering results, particularly with Houdini's wide array of data-handling tools.

We rendered visualizations of the synestia with a shading network informed by the 5-cluster GMM result, and compared this to an identical render with a custom shading network by the Advanced Visualization Lab at the National Center for Supercomputing Applications for "Imagine the Moon". We showed that with simple assumptions and the clustering result, it is possible to achieve a render similar in quality to a professional team of visualization designers. Furthermore, using our clustering and visualization pipeline, other scientifically-informed renders (e.g. segment and show distinct, meaningful clusters) can be compared in a fraction of the time. Renders simply of the particles' clustering assignment can help understand the context of the different physical structures and inform scientific discovery.

Our results are significant in that they help realize the context of 2D phase–space information in 3D position space in a relatively simple manner. Because Houdini is a visual effects software, it can provide benefits not found in scientific software such as (but not limited to) excellent camera and animation controls, general-purpose ability, high-quality renders, robustness from large user and developer base, etc. Though, Houdini will be best paired in tandem with scientific software due to their native ability to read specific data formats, and there have been custom software that best utilize the two for visualizations such as `ytini` (Naiman et al., 2017).

Cinematic astrophysical data visualization is an underdeveloped field in the literature, and its practices are not widely adopted by the astronomical community. By establishing such a literature, we hope to equip astronomers with the tools, skills, and knowledge to develop their own visualizations for publications, public outreach, prototype testing, etc.

We suggest future endeavors focus on developing tools for multi-timestep data, where cluster assignments are made temporally. Specifically, tools that track and visualize particles by their cluster assignment in each time step would allow researchers to see not only how clusters change and evolve over time, but how their members do as well. This is invaluable information, and can lead to a better understanding of physical processes acting on both small and large scales. Lastly, future work which incorporates extrinsic audience perceptual metrics can help visualization designers better understand how factors such as lighting, occlusion, and color influence affect an audience's perception of the spatial and scientific reality of a dataset.

Software: Houdini,[24] Jupyter (Kluyver et al., 2016), Matplotlib (Hunter, 2007), NumPy (Oliphant, 2015), pandas (McKinney, 2010; Pandas development team, 2020), scikit-learn (Pedregosa et al., 2011), SciPy (Virtanen et al., 2020)

CRediT authorship contribution statement

P.D. Aleo: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization. **S.J. Lock:** Validation, Formal analysis, Investigation, Resources, Writing - original draft, Writing - review & editing, Funding acquisition. **D.J. Cox:** Conceptualization, Writing - review & editing, Supervision, Project administration, Funding acquisition. **S.A. Levy:** Software, Formal analysis, Resources, Writing - review & editing, Visualization. **J.P. Naiman:** Formal analysis, Writing - review & editing. **A.J. Christensen:** Conceptualization, Software, Validation, Formal analysis, Resources, Writing - review & editing, Visualization, Supervision. **K. Borkiewicz:** Conceptualization, Software, Validation, Formal analysis, Resources, Writing - review & editing, Visualization, Supervision. **R. Patterson:** Validation, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank all the additional members of the Advanced Visualization Lab at the National Center for Supercomputing Applications for their invaluable support, input, and insight: Jeffrey Carpenter and Colter Wehmeier. The authors also thank Matthew J. Turk for his vital role in overseeing many aspects of this work, as well as the anonymous referee for helpful comments. Additionally, this research was made possible, in part, by funds from the 2019 Fiddler Innovation Fellowship generously supported by Jerry Fiddler, USA. SJL acknowledges funding from the Caltech Division of Geological and Planetary Sciences, USA and NSF, USA through grants EAR-1947614 and EAR-1725349.

Appendix A. Other clustering algorithms

Because the choice and interpretation of various clustering algorithms are vital to producing structurally-meaningful results, we provide a quick overview of several widely-used methods.

A.1. Kmeans

The first of many unsupervised clustering algorithms available to use in `Estra` is Kmeans,[25] through the `sklearn.cluster` module. As a whole, Kmeans is likely the most popular of the array of the clustering methods, perhaps due to its scalability to large sample sizes. After specifying a desired number of n-clusters (which itself is either a known prior, or an estimated one) the data will be split into n-groups of equal variance by minimizing the inertia (also known as "within-cluster sum-of-squares") criterion. More specifically, it is the mean values of each cluster, referred to as centroids, that minimize the inertia. A minimization of the inertia is optimizing how internally coherent the clusters are, such that each member within a particular cluster is most similar to its members and most dissimilar to members outside that particular cluster. Thus, generally speaking, Kmeans is useful for relatively few number of clusters with approximately even cluster sizes of flat geometry, where no explicit structure relates one cluster to another.

The Kmeans cost function is given by

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(\|x_i - \mu_j\|^2), \tag{6}$$

where the standard L_2 Euclidean distance is minimized between each ith cluster data point x_i and the collection of centroids μ_j in the set C. Once the algorithm satisfies some stopping criterion (say, some maximum number of iterations is completed or no data points change their cluster assignment between iterations),

[24] https://www.sidefx.com/products/houdini/.

[25] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

the output will be dataset labels of all data points for user-selected K-clusters. This convergence may be a local optimum, as some randomized starting centroids will ultimately provide better results than others. We suggest running this algorithm multiple times and seeding with different initial guesses.

As with any algorithm, there are downsides to Kmeans and situations in which other algorithms are better suited. One such drawback is for data with irregularly shaped manifolds, because a critical underlying assumption of the inertia criterion is that the clusters are convex and isotropic. Although the algorithm will reach a convergence, it is unlikely then that this particular result will correspond to a physical meaning; it will more likely be a mathematical artifact. Another case where Kmeans suffers is in reconciling its Euclidean distance measurements when higher dimensions are involved—namely the "curse of dimensionality". When plotting very high dimensional data in a lower dimensional space, the distances between points start to lose meaning; by nature of not being able to aptly express the true nature of the data in all of its dimensionality, some variance is lost. Thus, as Kmeans utilizes standard Euclidean distance measurements, these distances become inflated in higher-dimensional space. In this case, then, we suggest first running some dimensionality reduction such as Principal Component Analysis (PCA), Random Forest (RF), t-Distributed Stochastic Neighbor Embedding (t-SNE), Non-negative Matrix Factorization (NMF), etc.

A.2. DBSCAN

DBSCAN[26] (Density-Based Spatial Clustering of Applications with Noise) is quite different from Kmeans and other clustering algorithms in that it does not care about the shape of the clusters (Ester et al., 1996). Instead, the main focus of the algorithm is to separate areas of high density from low density. DBSCAN achieves this by using two main parameters: ϵ(eps) and *min_samples* (MinPts in Ester et al. (1996)). The ϵ parameter is used to assign a maximum distance for which two points can be considered to be within the same neighborhood, and the *min_samples* parameter is the number of datapoints needed in a neighborhood for a datapoint to be labeled a "core point". Put another way, it assigns a core point p if there are $\geq$ *min_samples* points within a given distance ϵ of p itself. Then, if another point q is contained within a distance ϵ of p, the point q is known to be "directly reachable" to be p (where "directly reachable" only applies to core points). Only points directly reachable to a core point, including both non-core and the core points themselves, compose a single cluster; all points that fall outside of the range for which any one core point is directly reachable is deemed an "outlier" or "noise point", and is not considered part of any cluster. Note that the non-core points directly reachable to only $\leq$ *min_samples* core points are called "edge points". However, there is one more aspect in deciding which points belong to which cluster: "density connectedness". If there is another point o such that any two points p, q are both directly reachable from it, the points p and q are "density-connected". Thus for any particular cluster, all points contained within it are (1) mutually density-connected and (2) directly reachable to at least one core point (making this a region of "higher density").

As a density-based clustering algorithm, DBSCAN offers some critical advantages over Kmeans and various traditional clustering methods. One such benefit is that the number of clusters is not required to be known *a priori*; instead, a knowledge of the dataset on how to best set the values for ϵ and *min_samples* is suggested, but not required. Further, as aforementioned, it can find arbitrarily shaped clusters, even those which are not linearly separable (non-flat geometry), in addition to finding unevenly sized clusters, as long as they are of the same relative density. Inherent to its formulation, DBSCAN is robust to outliers as well, as it factors in noise into its cluster finding process.

The main disadvantage of DBSCAN is in regard to its memory consumption when large data samples are involved. In fact, the `sklearn.cluster` Python package for DBSCAN involves the worst-case memory storage of $\mathcal{O}(n^2)$ floats when a full pairwise similarity matrix is used.[27] The pairwise similarity matrix is used in the case where its default method of *k*d-trees or ball-trees is not applicable, such as for sparse matrices; otherwise, the user can choose their preferred nearest neighbors module to compute pointwise distances and find nearest neighbors. Another main disadvantage lies with the curse of dimensionality, as before. DBSCAN uses *k*d-trees or ball-trees for its nearest neighbors search, and in high dimensional space, these methods (particularly *k*d-trees) are not well suited for efficiently finding such nearest neighbors. Often times a simple exhaustive search is as efficient, and it is better to use more approximate nearest neighbor methods. Further, as pointed out above, higher dimensions render the Euclidean distance metric for all practical purposes useless, and DBSCAN uses Euclidean distance as a measure for determining which points are "directly reachable" to each other. And lastly, if there is a wide range of densities within the data, the particular chosen values of ϵ and *min_samples* may not be appropriate when scaled to all clusters.

A.3. Variational Bayesian Gaussian mixture model

Variational Bayesian Gaussian Mixture Model is a variant of GMM that utilizes variational (Bayesian) inference (Jordan et al., 1999; Wainwright and Jordan, 2008), instead of EM, to fit the model (Attias, 2000). This is known as variational Bayesian estimation of a Gaussian mixture model, a form of approximate posterior inference.

We desire to know the true posterior of the distribution, which from Bayes' Rule is

$$p(\mathbf{z}|\mathbf{x},\alpha) = \frac{p(\mathbf{z},\mathbf{x}|\alpha)}{\int_Z p(\mathbf{z},\mathbf{x}|\alpha)} = \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})}, \tag{7}$$

where $\mathbf{z} = z_{1:m}$ are our hidden, or latent, variables, $\mathbf{x} = x_{1:n}$ are our observations, or data (e.g. the particles of the simulation), and α are the fixed additional parameters (though, if the parameters are lumped into $\mathbf{z}$, then α becomes the hyperparameters). Computing the posterior distribution is what is known to be an inference problem.

Finding the true posterior, or, as will be the case, finding an approximate posterior, is important for predictive distribution (given the data $\mathbf{x}$, compute the conditional probability of a new observation x^*, denoted as $p(x^*|\mathbf{x})$), finding modes, investigating the posterior over hidden variables, etc. Unfortunately, computing the normalization constant of the posterior for many complex models, including Gaussian mixture, is often analytically intractable. One could use Markov-Chain Monte Carlo (MCMC) sampling (Hastings, 1970; Gelfand and Smith, 1990) to find the true posterior, but with many parameters, convergence can be prohibitively slow, and it does not scale to large data as well as variational inference.

A simpler, tractable family of distributions $\mathcal{D}$ over the latent variables $\mathbf{z}$ is formed with its own collection of variational parameters ν, i.e. $q(\mathbf{z}|\nu)$. These variational parameters are chosen so that q becomes a proxy for the desired posterior. In essence,

[26] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[27] See DBSCAN's user guide in scikit-learn's documentation.

$p(\mathbf{z}|\mathbf{x}, \alpha) \approx q(\mathbf{z}|\nu, \alpha)$. To achieve this, one finds the particular family member that minimizes the Kullback–Leibler (KL) divergence with respect to the exact posterior

$$q^*(\mathbf{z}|\nu, \alpha) = \underset{q(\mathbf{z}|\nu,\alpha)\in\mathcal{D}}{\operatorname{argmin}} \operatorname{KL}(q(\mathbf{z}|\nu, \alpha)\|p(\mathbf{z}|\mathbf{x}, \alpha)), \tag{8}$$

where $q^*(\cdot)$ is the optimized member of each family, and KL is defined to be

$$\operatorname{KL}(q\|p) = \int_z \log \frac{q(z)}{p(z|x)} = \mathbb{E}\Big[\log \frac{q(z)}{p(z|x)}\Big]. \tag{9}$$

Because the KL divergence is an asymmetric measure of the difference between two distributions p and q, its minimization will ensure that p is most similar to q. Thus, one can use the approximate conditional $q^*(\mathbf{z}|\nu, \alpha)$ as a best guess to the posterior $p(\mathbf{z}|\mathbf{x}, \alpha)$, and subsequently replace the posterior with the approximation. In all, variational inference rewrites a standard inference problem as an *optimization* problem. For a more rigorous mathematical justification and explanation, see Blei et al. (2017).

In the `BayesianGaussianMixture` class, two types of weighting schemes are available to be specified by the user: (1) a finite mixture model using Dirichlet Distribution, and (2) an infinite mixture model using Dirichlet Process (DP). The reason for implementing the Dirichlet distribution may be nebulous, so first consider the joint distribution $p(\mathbf{z}, \mathbf{x})$. From Bayes' Rule, as shown in Eq. (7), the posterior is proportional to the product of the prior of the latent variable (cluster), $p(\mathbf{z})$, with the conditional $p(\mathbf{x}|\mathbf{z})$. In a GMM, z_i is drawn independent and identically distributed (*i.d.d.*) from the multinomial (or categorical) distribution, and the conditional $x_i|z_i$ is drawn from a Gaussian distribution. Because the Dirichlet distribution is the conjugate prior of the multinomial (and categorical) distribution, that means if the likelihood – the generative model – is multinomial (or categorical), then both the prior and the posterior are Dirichlet distributed. Then, with the form of the posterior known to be Dirichlet, it is easier to compute when using the variational Bayesian inference method.

From this point, all that is left is selecting between a finite mixture model with Dirichlet Distribution and a non-parametric infinite mixture model using Dirichlet Process (DP). DP (analogous to the Chinese Restaurant Process and Stick-breaking Process) is simply an "infinite-dimensional" Dirichlet defined by an infinite number of clusters and a concentration parameter. Although an infinite number of clusters are available only a finite number are used to generate the data. This enables the user to not have to pre-specify the number of clusters. In fact, this setting only requires the user to specify the concentration parameter and the upper-limit on the number of mixture components, and the value of the `weight_concentration_prior` will decide to use either all or only some of the components. The lower the value, the fewer components (some very close to zero); the higher the value, the more active mixture components (as well as more uniform).

Appendix B. Renders with perceptually-uniform temperature colormaps

One of the decisions a visualization designer has to make when developing a visualization is what type of color map to use. A poorly chosen color map can trick human eyes into seeing non-existent features, mixing features, or missing features altogether. One such example is a rainbow color map (Borland and Taylor II, 2007; Moreland, 2016). A perceptually-uniform colormap takes human visual perception knowledge and gives readers a correct sense on the image intensities regardless of the display. The main motivating factor is biological; the human eye is more sensitive to brightness than hue (Borkiewicz et al., 2019a). Thus, any subtle change in brightness is more readily recognized. For example, the blackbody colormap from Moreland (2016) has colors which are based on those from blackbody radiation, but are not exact according to its wavelength. Instead, they increase in brightness at a constant rate, and the luminance is perceptually linear (in CIELAB color space).

Because the synestia is emissive, we rendered the image seen in Fig. 16 using a perceptually uniform colormap named *afmhot_us* from the `ehtplot`[28] library developed for Event Horizon Telescope Collaboration et al. (2019). This specific color map is symmetrized with linearity in lightness J' as defined by the CAM02-UCS color appearance model introduced by Luo et al. (2006). A linearity of J' values is a good approximation of uniform colormaps, and is the working definition of Perceptually Uniform Sequential colormaps by matplotlib.[29]

As is evident, Figs. 16(a), 16(b) are more gradual in their color change, and the dust ring appears to blend in more to its central region than those in Fig. 11. Although interpretation is subjective, overall it is harder to distinguish here between the different structures of the synestia as seen in Fig. 15, compared to Fig. 11(a).

The only appreciable difference between renders in Fig. 16(a) and 16(b) is the pink appearing highlights in the left side of the bulge, which marginally differentiates this isentropic component from the rest of the bulge. This is because in this example, the perceptually-uniform colormap makes it more difficult to discern the synestia in Figs. 16(a), 16(b). Because of the linearity in brightness and a narrower color palette range, the color value of the bulge is more similar to the image background, making it more difficult to distinguish the demarcation between source and background. This is likewise for the dusty ring in the plane perpendicular to its rotation axis compared to the background.

CIELAB color space, a device-independent model which describes all colors visible to the human eye, has three related color spaces: CIE76, CIE94, and CIEDE2000. All three of these color spaces define a slightly different metric for color difference called ΔE^*_{00} (Luo et al., 2001), which attempts to quantify how noticeable two colors are based on knowledge that the human eye is more sensitive to some colors than others. However, in all formulae of ΔE^*_{00}, Fig. 11(a) has approximately equal or greater values for all bulge-background and ring-background comparisons than Figs. 16(a), 16(b). Because these formulae are based on Euclidean distance measurements in color space, the greater the value between two colors represents the greater the color difference, and thus the more perceptually different the two colors are.

For instance, an example pixel in the brightest part of the bulge ("reference") and background ("sample") of Fig. 11(a) has $(R, G, B) = (209, 214, 192)$ and $(R, G, B) = (80, 34, 1)$, respectively[30]. According to the most recent CIEDE2000 definition[31] given by

$$\Delta E^*_{00} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \left(\frac{\Delta C'}{k_C S_C}\right)\left(\frac{\Delta H'}{k_H S_H}\right)}, \tag{10}$$

we find $\Delta E^*_{00} = 68.6$. Meanwhile, Fig. 16(b), has $(R, G, B) = (246, 208, 142)$ and $(R, G, B) = (136, 46, 12)$, respectively, which results in $\Delta E^*_{00} = 51.7$. Hence, it is easier to discern the bulge and background in Fig. 11(a) as opposed to in Fig. 16(b).

28 https://github.com/eventhorizontelescope/ehtplot.

29 https://matplotlib.org/tutorials/colors/colormaps.html.

30 Before applying the formula, (R, G, B) is converted to CIE-(L^*, a^*, b^*) coordinates.

31 See Sharma et al. (2005) for info on the CIEDE2000 color difference formulae and explanation.

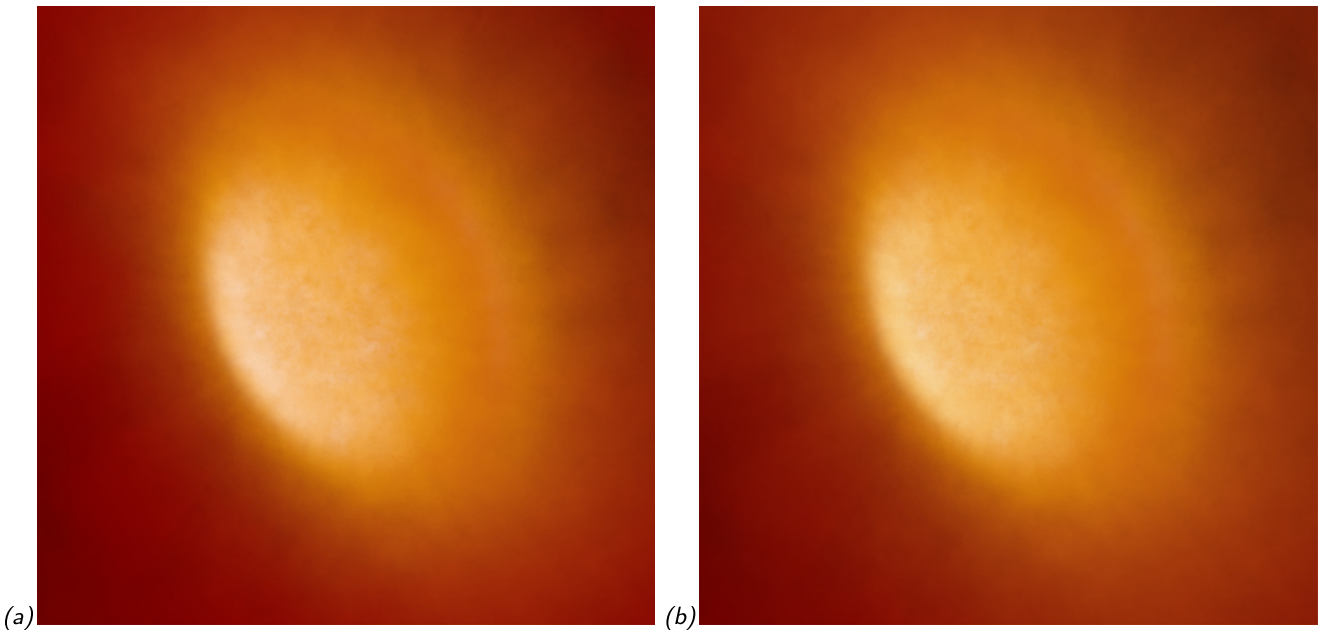

Fig. 16. Same as Fig. 11, except this applies the same perceptually uniform colormap used in the black hole imaging by the Event Horizon Telescope: *afmhot_us* (Event Horizon Telescope Collaboration et al., 2019). *(a)* uses the 5-cluster GMM results with the simplified `Estra` shader network, and *(b)* uses the custom AVL shader without clustering.

Although using the clustering-informed results on a perceptually-uniform colormap does not appear to be easily distinguishable in this case, it does in Fig. 11(a) where a non-perceptually uniform colormap is used.

In deciding the "best" render for this work, we argue that one which emphasizes by features of the inner regions of the terrestrial synestia from the extended disk-like regions is optimal. From the calculation of ΔE^*_{00}, this is clearly Fig. 11(a). However, each dataset is different, and a perceptually uniform colormap can be a quick and useful starting place for visualization teams or scientists.

References

Arroio, A., 2010. Context based learning: A role for cinema in science education. Sci. Educ. Int. 21 (3), 131–143.

Attias, H., 2000. A variational Bayesian framework for graphical models. In: Advances in Neural Information Processing Systems 12. MIT Press, pp. 209–215.

Ball, N.M., Brunner, R.J., 2010. Data mining and machine learning in astronomy. Internat. J. Modern Phys. D 19 (7), 1049–1106. doi:10.1142/S0218271810017160, arXiv:0906.2173.

Barnes, D.G., Fluke, C.J., 2008. Incorporating interactive three-dimensional graphics in astronomy research papers. New Astron. 13, 599–605. doi:10.1016/j.newast.2008.03.008, arXiv:0709.2734.

Baron, D., 2019. Machine learning in astronomy: a practical overview. arXiv e-prints, arXiv:1904.07248.

Berger, M.J., Oliger, J., 1984. Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys. 53, 484–512. doi:10.1016/0021-9991(84)90073-1.

Blei, D.M., Kucukelbir, A., McAuliffe, J.D., 2017. Variational inference: A review for statisticians. J. Amer. Statist. Assoc. 112 (518), 859–877. doi:10.1080/01621459.2017.1285773.

Borkiewicz, K., Christensen, A., Kostis, H.-N., Shirah, G., Wyatt, R., 2019a. Cinematic scientific visualization: The art of communicating science. In: ACM SIGGRAPH 2019 Courses, SIGGRAPH '19. ACM, New York, NY, USA, pp. 5:1–5:273. doi:10.1145/3305366.3328056, URL http://doi.acm.org/10.1145/3305366.3328056.

Borkiewicz, K., Christensen, A., Stone, J.E., 2017. Communicating science through visualization in an age of alternative facts. In: ACM SIGGRAPH 2017 Courses, SIGGRAPH '17. ACM, New York, NY, USA, pp. 8:1–8:204. doi:10.1145/3084873.3084935, URL http://doi.acm.org/10.1145/3084873.3084935.

Borkiewicz, K., Naiman, J.P., Lai, H., 2019b. Cinematic visualization of multiresolution data: Ytini for adaptive mesh refinement in houdini. Astron. J. 158 (1), 10. doi:10.3847/1538-3881/ab1f6f.

Borland, D., Taylor II, R.M., 2007. Rainbow color map (still) considered harmful. IEEE Comput. Graph. Appl. 27 (2), 14–17. doi:10.1109/MCG.2007.323435.

Burke, C.J., Aleo, P.D., Chen, Y.-C., Liu, X., Peterson, J.R., Sembroski, G.H., Lin, J.Y.-Y., 2019. Deblending and classifying astronomical sources with mask r-CNN deep learning. Mon. Not. R. Astron. Soc. doi:10.1093/mnras/stz2845, arXiv:1908.02748.

Cameron, A.G.W., Ward, W.R., 1976. The origin of the moon. In: Lunar and Planetary Science Conference. In: Lunar and Planetary Science Conference, vol. 7, p. 120.

Cawthon, N., Moere, A.V., 2007. The effect of aesthetic on the usability of data visualization. In: Proceedings of the 11th International Conference Information Visualization, IV '07. IEEE Computer Society, Washington, DC, USA, pp. 637–648. doi:10.1109/IV.2007.147, URL https://doi.org/10.1109/IV.2007.147.

Cox, D., 2008. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. Leonardo 41 (4), 233–243.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1), 1–38, URL http://www.jstor.org/stable/2984875.

Dubeck, L., Moshier, S., Boss, J., 1994. Fantastic Voyages : Learning Science Through Science Fiction Films. American Institute of Physics New York, xiv, 327 p.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96. AAAI Press, pp. 226–231, URL http://dl.acm.org/citation.cfm?id=3001460.3001507.

Event Horizon Telescope Collaboration, Akiyama, K., Alberdi, A., Alef, W., Asada, K., Azulay, R., Baczko, A.-K., Ball, D., Baloković, M., Barrett, J., Bintley, D., Blackburn, L., Boland, W., Bouman, K.L., Bower, G.C., Bremer, M., Brinkerink, C.D., Brissenden, R., Britzen, S., Broderick, A.E., Broguiere, D., Bronzwaer, T., Byun, D.-Y., Carlstrom, J.E., Chael, A., Chan, C.-k., Chatterjee, S., Chatterjee, K., Chen, M.-T., Chen, Y., Cho, I., Christian, P., Conway, J.E., Cordes, J.M., Crew, G.B., Cui, Y., Davelaar, J., De Laurentis, M., Deane, R., Dempsey, J., Desvignes, G., Dexter, J., Doeleman, S.S., Eatough, R.P., Falcke, H., Fish, V.L., Fomalont, E., Fraga-Encinas, R., Freeman, W.T., Friberg, P., Fromm, C.M., Gómez, J.L., Galison, P., Gammie, C.F., García, R., Gentaz, O., Georgiev, B., Goddi, C., Gold, R., Gu, M., Gurwell, M., Hada, K., Hecht, M.H., Hesper, R., Ho, L.C., Ho, P., Honma, M., Huang, C.-W.L., Huang, L., Hughes, D.H., Ikeda, S., Inoue, M., Issaoun, S., James, D.J., Jannuzi, B.T., Janssen, M., Jeter, B., Jiang, W., Johnson, M.D., Jorstad, S., Jung, T., Karami, M., Karuppusamy, R., Kawashima, T., Keating, G.K., Kettenis, M., Kim, J.-Y., Kim, J., Kim, J., Kino, M., Koay, J.Y., Koch, P.M., Koyama, S., Kramer, M.,

Kramer, C., Krichbaum, T.P., Kuo, C.-Y., Lauer, T.R., Lee, S.-S., Li, Y.-R., Li, Z., Lindqvist, M., Liu, K., Liuzzo, E., Lo, W.-P., Lobanov, A.P., Loinard, L., Lonsdale, C., Lu, R.-S., MacDonald, N.R., Mao, J., Markoff, S., Marrone, D.P., Marscher, A.P., Martí-Vidal, I., Matsushita, S., Matthews, L.D., Medeiros, L., Menten, K.M., Mizuno, Y., Mizuno, I., Moran, J.M., Moriyama, K., Moscibrodzka, M., Müller, C., Nagai, H., Nagar, N.M., Nakamura, M., Narayan, R., Narayanan, G., Natarajan, I., Neri, R., Ni, C., Noutsos, A., Okino, H., Olivares, H., Ortiz-León, G.N., Oyama, T., Özel, F., Palumbo, D.C.M., Patel, N., Pen, U.-L., Pesce, D.W., Piétu, V., Plambeck, R., PopStefanija, A., Porth, O., Prather, B., Preciado-López, J.A., Psaltis, D., Pu, H.-Y., Ramakrishnan, V., Rao, R., Rawlings, M.G., Raymond, A.W., Rezzolla, L., Ripperda, B., Roelofs, F., Rogers, A., Ros, E., Rose, M., Roshanineshat, A., Rottmann, H., Roy, A.L., Ruszczyk, C., Ryan, B.R., Rygl, K.L.J., Sánchez, S., Sánchez-Arguelles, D., Sasada, M., Savolainen, T., Schloerb, F.P., Schuster, K.-F., Shao, L., Shen, Z., Small, D., Sohn, B.W., SooHoo, J., Tazaki, F., Tiede, P., Tilanus, R.P.J., Titus, M., Toma, K., Torne, P., Trent, T., Trippe, S., Tsuda, S., van Bemmel, I., van Langevelde, H.J., van Rossum, D.R., Wagner, J., Wardle, J., Weintroub, J., Wex, N., Wharton, R., Wielgus, M., Wong, G.N., Wu, Q., Young, K., Young, A., Younsi, Z., Yuan, F., Yuan, Y.-F., Zensus, J.A., Zhao, G., Zhao, S.-S., Zhu, Z., Algaba, J.-C., Allardi, A., Amestica, R., Anczarski, J., Bach, U., Baganoff, F.K., Beaudoin, C., Benson, B.A., Berthold, R., Blanchard, J.M., Blundell, R., Bustamente, S., Cappallo, R., Castillo-Domínguez, E., Chang, C.-C., Chang, S.-H., Chang, S.-C., Chen, C.-C., Chilson, R., Chuter, T.C., Córdova Rosado, R., Coulson, I.M., Crawford, T.M., Crowley, J., David, J., Derome, M., Dexter, M., Dornbusch, S., Dudevoir, K.A., Dzib, S.A., Eckart, A., Eckert, C., Erickson, N.R., Everett, W.B., Faber, A., Farah, J.R., Fath, V., Folkers, T.W., Forbes, D.C., Freund, R., Gómez-Ruiz, A.I., Gale, D.M., Gao, F., Geertsema, G., Graham, D.A., Greer, C.H., Grosslein, R., Gueth, F., Haggard, D., Halverson, N.W., Han, C.-C., Han, K.-C., Hao, J., Hasegawa, Y., Henning, J.W., Hernández-Gómez, A., Herrero-Illana, R., Heyminck, S., Hirota, A., Hoge, J., Huang, Y.-D., Impellizzeri, C.M.V., Jiang, H., Kamble, A., Keisler, R., Kimura, K., Kono, Y., Kubo, D., Kuroda, J., Lacasse, R., Laing, R.A., Leitch, E.M., Li, C.-T., Lin, L.C.C., Liu, C.-T., Liu, K.-Y., Lu, L.-M., Marson, R.G., Martin-Cocher, P.L., Massingill, K.D., Matulonis, C., McColl, M.P., McWhirter, S.R., Messias, H., Meyer-Zhao, Z., Michalik, D., Montaña, A., Montgomerie, W., Mora-Klein, M., Muders, D., Nadolski, A., Navarro, S., Neilsen, J., Nguyen, C.H., Nishioka, H., Norton, T., Nowak, M.A., Nystrom, G., Ogawa, H., Oshiro, P., Oyama, T., Parsons, H., Paine, S.N., Peñalver, J., Phillips, N.M., Poirier, M., Pradel, N., Primiani, R.A., Raffin, P.A., Rahlin, A.S., Reiland, G., Risacher, C., Ruiz, I., Sáez-Madaín, A.F., Sassella, R., Schellart, P., Shaw, P., Silva, K.M., Shiokawa, H., Smith, D.R., Snow, W., Souccar, K., Sousa, D., Sridharan, T.K., Srinivasan, R., Stahm, W., Stark, A.A., Story, K., Timmer, S.T., Vertatschitsch, L., Walther, C., Wei, T.-S., Whitehorn, N., Whitney, A.R., Woody, D.P., Wouterloot, J.G.A., Wright, M., Yamaguchi, P., Yu, C.-Y., Zeballos, M., Zhang, S., Ziurys, L., 2019. First M87 event horizon telescope results. I. The shadow of the supermassive black hole. Astrophys. J. Lett. 875 (1), L1. doi:10.3847/2041-8213/ab0ec7, arXiv:1906.11238.

Gelfand, A.E., Smith, A.F.M., 1990. Sampling-based approaches to calculating marginal densities. J. Amer. Statist. Assoc. 85 (410), 398–409. doi:10.1080/01621459.1990.10476213, URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1990.10476213, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1990.10476213.

Gingold, R.A., Monaghan, J.J., 1977. Smoothed particle hydrodynamics - theory and application to non-spherical stars. Mon. Not. R. Astron. Soc. 181, 375–389. doi:10.1093/mnras/181.3.375.

Goodman, A.A., 2012. Principles of high-dimensional data visualization in astronomy. Astron. Nachr. 333, 505. doi:10.1002/asna.201211705, arXiv:1205.4747.

Hartmann, W.K., Davis, D.R., 1975. Satellite-sized planetesimals and lunar origin. Icarus 24 (4), 504–515. doi:10.1016/0019-1035(75)90070-6.

Hassan, A., Fluke, C.J., 2011. Scientific visualization in astronomy: Towards the petascale astronomy era. Publ. Astron. Soc. Aust. 28 (2), 150–170. doi:10.1071/AS10031.

Hastings, W.K., 1970. Monte Carlo Sampling methods using Markov chains and their applications. Biometrika 57 (1), 97–109, URL http://www.jstor.org/stable/2334940.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9 (3), 90–95. doi:10.1109/MCSE.2007.55.

Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K., 1999. An introduction to variational methods for graphical models. Mach. Learn. 37, 183–233.

Kähler, R., Cox, D., Patterson, R., Levy, S., Hege, H.C., Abel, T., 2002. Rendering the first star in the universe: A case study. In: Proceedings of the Conference on Visualization '02, VIS '02. IEEE Computer Society, Washington, DC, USA, pp. 537–540, URL http://dl.acm.org/citation.cfm?id=602099.602191.

Kent, B.R., 2013. Visualizing astronomical data with blender. Publ. Astron. Soc. Pac. 125 (928), 731–748. doi:10.1086/671412.

Kent, B.R., 2015. 3D Scientific Visualization with Blender.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., development team, J., 2016. Jupyter notebooks - a publishing format for reproducible computational workflows. In: Loizides, F., Scmidt, B. (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, pp. 87–90, URL https://eprints.soton.ac.uk/403913/.

Liu, E., Llamas, I., Cañada, J., Kelly, P., 2019. Cinematic rendering in ue4 with real-time ray tracing and denoising. In: Haines, E., Akenine-Möller, T. (Eds.), Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs. Apress, Berkeley, CA, pp. 289–319. doi:10.1007/978-1-4842-4427-2_19.

Lock, S.J., Stewart, S.T., 2017. The structure of terrestrial bodies: Impact heating, corotation limits, and synestias. Journal of Geophysical Research: Planets 122 (5), 950–982. doi:10.1002/2016JE005239, URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016JE005239, arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2016JE005239.

Lock, S.J., Stewart, S.T., Petaev, M.I., Leinhardt, Z., Mace, M.T., Jacobsen, S.B., Cuk, M., 2018. The origin of the moon within a terrestrial synestia. Journal of Geophysical Research (Planets) 123, 910–951. doi:10.1002/2017JE005333, arXiv:1802.10223.

Lowe, R., 2003. Animation and learning: selective processing of information in dynamic graphics. In: External and Internal Representations in Multimedia Learning. Learn. Instr. 13 (2), 157–176. doi:10.1016/S0959-4752(02)00018-X, URL http://www.sciencedirect.com/science/article/pii/S095947520200018X.

Luo, M.R., Cui, G., Li, C., 2006. Uniform colour spaces based on ciecam02 colour appearance model. Color Res. Appl. 31 (4), 320–330. doi:10.1002/col.20227, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/col.20227.

Luo, M.R., Cui, G., Rigg, B., 2001. The development of the CIE 2000 colour-difference formula: CIEDE2000. Color Res. Appl. 26 (5), 340–350. doi:10.1002/col.1049, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/col.1049, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/col.1049.

Ma, K.-L., 2007. Machine learning to boost the next generation of visualization technology. IEEE Comput. Graph. Appl. 27 (5), 6–9. doi:10.1109/MCG.2007.129, URL https://doi.org/10.1109/MCG.2007.129.

McKinney, W., 2010. Data structures for statistical computing in python. In: Stéfan van der Walt and Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, p. 56–61, http://dx.doi.org/10.25080/Majora-92bf1922-00a.

Milosavljevic, M., Aleo, P.D., Hinkel, N.R., Vikalo, H., 2018. Blind nucleosynthetic source discovery in astronomical elemental abundance data. arXiv e-prints, arXiv:1809.02660.

Moreland, K., 2016. Why we use bad color maps and what you can do about it. Electron. Imaging 2016, 1–6. doi:10.2352/ISSN.2470-1173.2016.16.HVEI-133.

Naiman, J.P., 2016. AstroBlend: An astrophysical visualization package for blender. Astron. Comput. 15, 50–60. doi:10.1016/j.ascom.2016.02.002, arXiv:1602.03178.

Naiman, J.P., Borkiewicz, K., Christensen, A.J., 2017. Houdini for astrophysical visualization. Publ. Astron. Soc. Pac. 129 (5), 058008. doi:10.1088/1538-3873/aa51b3, arXiv:1701.01730.

Nakajima, M., Stevenson, D.J., 2015. Melting and mixing states of the earth's mantle after the moon-forming impact. Earth Planet. Sci. Lett. 427, 286–295. doi:10.1016/j.epsl.2015.06.023, URL http://www.sciencedirect.com/science/article/pii/S0012821X15003805.

Oliphant, T.E., 2015. Guide to NumPy, second ed. CreateSpace Independent Publishing Platform, North Charleston, SC, USA.

O'Rourke, M., 1998. Principles of Three-Dimensional Computer Animation: Modeling, Rendering, and Animating with 3d Computer Graphics, second ed. W. W. Norton & Co., Inc., USA.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Pesenson, M.Z., Pesenson, I.Z., McCollum, B., 2010. The data big bang and the expanding digital universe: high-dimensional, complex and massive data sets in an inflationary epoch. Adv. Astron. 2010, 350891. doi:10.1155/2010/350891, arXiv:1003.0879.

Price, D.J., 2007. Splash: An interactive visualisation tool for smoothed particle hydrodynamics simulations. Publ. Astron. Soc. Aust. 24, 159–173. doi:10.1071/AS07022, arXiv:0709.0832.

Punzo, D., van der Hulst, J.M., Roerdink, J.B.T.M., Fillion-Robin, J.C., Yu, L., 2017. Slicerastro: A 3-d interactive visual analytics tool for HI data. Astron. Comput. 19, 45–59. doi:10.1016/j.ascom.2017.03.004, arXiv:1703.06651.

Punzo, D., van der Hulst, J.M., Roerdink, J.B.T.M., Oosterloo, T.A., Ramatsoku, M., Verheijen, M.A.W., 2015. The role of 3-D interactive visualization in blind surveys of H I in galaxies. Astron. Comput. 12, 86–99. doi:10.1016/j.ascom.2015.05.004, arXiv:1505.06976.

Raymond, S.N., Izidoro, A., Morbidelli, A., 2018. Solar system formation in the context of extra-solar planets. arXiv e-prints, arXiv:1812.01033.

Reynolds, D.A., 2009. Gaussian mixture models. In: Encyclopedia of Biometrics.

Rivi, M., Gheller, C., Dykes, T., Krokos, M., Dolag, K., 2014. GPU Accelerated particle visualization with splotch. Astron. Comput. 5, 9–18. doi:10.1016/j.ascom.2014.03.001, URL http://www.sciencedirect.com/science/article/pii/S2213133714000109.

Scheufele, D.A., Krause, N.M., 2019. Science audiences, misinformation, and fake news. Proc. Natl. Acad. Sci. 116 (16), 7662–7669. doi:10.1073/pnas.1805871115, URL https://www.pnas.org/content/116/16/7662, https://www.pnas.org/content/116/16/7662.full.pdf.

Sharma, G., Wu, W., Dalal, E.N., 2005. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. Color Res. Appl. 30 (1), 21–30. doi:10.1002/col.20070, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/col.20070.

Shih, J.Y., Borkiewicz, K., Christensen, A., Cox, D., 2019. Interactive cinematic scientific visualization in unity. In: ACM SIGGRAPH 2019 Posters, SIGGRAPH '19. ACM, New York, NY, USA, pp. 69:1–69:2. doi:10.1145/3306214.3338588, URL http://doi.acm.org/10.1145/3306214.3338588.

Springel, V., Yoshida, N., White, S.D.M., 2001. GADGET: a code for collisionless and gasdynamical cosmological simulations. New Astron. 6 (2), 79–117. doi:10.1016/S1384-1076(01)00042-2, arXiv:astro-ph/0003162.

Sterken, C., Manfroid, J., 1992. Astronomical Photometry, A Guide, Vol. 175. doi:10.1007/978-94-011-2476-8.

The pandas development team, 2020. Pandas-dev/pandas: Pandas. Zenodo, doi:10.5281/zenodo.3509134.

Turk, M.J., Smith, B.D., Oishi, J.S., Skory, S., Skillman, S.W., Abel, T., Norman, M.L., 2011. yt: A multi-code analysis toolkit for astrophysical simulation data. Astrophys. J. Suppl. 192, 9. doi:10.1088/0067-0049/192/1/9, arXiv:1011.3514.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S..., 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nat. Methods 17, 261–272. doi:10.1038/s41592-019-0686-2.

Wainwright, M.J., Jordan, M.I., 2008. Graphical models, exponential families, and variational inference. Found. Trends Mach. Learn. 1 (1–2), 1–305. doi:10.1561/2200000001.

Welbourne, D.J., Grant, W.J., 2016. Science communication on YouTube: Factors that affect channel and video popularity. Publ. Understanding Sci. 25 (6), 706–718. doi:10.1177/0963662515572068, PMID: 25698225.