STAG: Smart Tools and Applications in Graphics (2021)
P. Frosini, D. Giorgi, S. Melzi, and E. Rodolà (Editors)

Visual Analysis of Popping in Progressive Visualization

E. Waterink, J. Kosinka, and S. Frey

Bernoulli Institute, University of Groningen, the Netherlands

Abstract

Progressive visualization allows users to examine intermediate results while they are further refined in the background. This makes them increasingly popular when dealing with large data and computationally expensive tasks. The characteristics of how preliminary visualizations evolve over time are crucial for efficient analysis; in particular unexpected disruptive changes between iterations can significantly hamper the user experience. This paper proposes a visualization framework to analyze the refinement behavior of progressive visualization. We particularly focus on sudden significant changes between the iterations, which we denote as popping artifacts, in reference to undesirable visual effects in the context of level of detail representations in computer graphics. Our visualization approach conveys where in image space and when during the refinement popping artifacts occur. It allows to compare across different runs of stochastic processes, and supports parameter studies for gaining further insights and tuning the algorithms under consideration. We demonstrate the application of our framework and its effectiveness via two diverse use cases with underlying stochastic processes: adaptive image space sampling, and the generation of grid layouts.

CCS Concepts

• ***Human-centered computing*** → ***Visualization design and evaluation methods;***

1. Introduction

Datasets generated by simulations and experiments are becoming increasingly large and complex, requiring advanced visualization methods for their analysis. This means that a long time can pass until the final result is produced with standard approaches, which can significantly impede user exploration. Progressive visualizations help users analyze these large volumes of data by allowing them to examine intermediate results of computationally expensive problems without having to wait for the computation to complete. This paradigm means splitting long computations into a series of approximate results improving with time and conveying this progress. It addresses scalability problems, as analysts can keep their attention on the results of long analyses as they continuously arrive. Progressive visualization is becoming increasingly popular, with promising results across various scenarios [FFNS19, PMS*21, FP16], but it also comes with various challenges [TPB*19].

Smooth refinement without a large number of disruptive changes to the visualization would be desirable for a good user experience in the simultaneous analysis. We denote such large changes from one iteration to another as *popping artifacts*, in relation to undesirable visual effects occurring with level of detail representations [XESV97, Hop98, SS09]. Popping artifacts in progressive visualization are generally expected to happen early on in a refinement process, but depending on the method and its parameters they might also be introduced—surprisingly for a user—after longer periods of comparably minor changes. This disruptive behaviour is important to understand for experts and users of progressive visualization techniques.

In this work we propose a framework for the analysis of progressive visualization techniques with a focus on popping artifacts, which we consider to be our main contribution. It combines several aspects of the process which allow the user to gain insight into its stability. In particular, we identify and explicitly consider four characteristics shared among progressive visualization techniques, which provides the basis for our proposed framework. First, these approaches exhibit some kind of measure or metric to assess the progress, e.g., via an objective function for optimization and machine learning-based approaches. Showing them for instance as a chart can provide some insights regarding refinement characteristics as a whole, but popping artifacts are often not clearly visible (e.g., averaged out or not apparent due to overdraw). This is the main motivation for the proposed explicit detection of popping artifacts, which the following components are directly based on. Second, progressive visualization methods often employ stochastic processes, i.e., refinement behavior and results vary across runs (this is commonly seen in optimization and machine learning-based techniques, but is also often the case with adaptive sampling to avoid systematic biases). The analysis requires the consideration of multiple runs for comprehensive assessment of the properties. Third, progressive methods typically exhibit parameters with a strong impact on refinement behavior, and a respective analysis can greatly help in determining adequate choices. Fourth, in particular in techniques for scientific visualization like volume rendering, the absolute spatial

DOI: 10.2312/stag.20211485

location, i.e., where something happens, is of high relevance. In this work, we demonstrate our approach via two concrete stochastic iterative refinement processes: 1) an adaptive image space sampling scheme simulating a ray tracer application which iteratively samples pixels, and 2) a process of tile grid generation for placing similar objects close to each other.

In Sec. 2 we give an overview of (different types of) progressive visualization. Sec. 3 introduces our framework for the visual analysis of popping in progressive visualization. Then, we describe the two use cases in detail in Sec. 4 and Sec. 5, respectively, and discuss how the generic visualization applies to them as well as some dedicated visualizations. Finally, we conclude this work in Sec. 6 and provide directions for future work.

2. Related Work

Rosenbaum and Schumann [RS09] identify a high potential of progressive refinement far across a large variety of application contexts. Zgraggen et al. [ZGC*17] study how progressive visualizations affect users in exploratory scenarios. They find that users perform equally well with either instantaneous (a hypothetical condition where results are shown almost immediately) or progressive visualizations in key metrics, such as insight discovery rates and dataset coverage, while blocking visualizations have detrimental effects. Fisher et al. [FPDs12, FDK12] have explored whether interaction techniques presenting query results from only incremental samples in a database scenario are sufficiently trustworthy for analysts. Stolper et al. [SPG14] define design goals for both the algorithms and visualizations in progressive visual analytics systems as well as an example progressive visual analytics system (Progressive Insights) for analyzing common patterns in a collection of event sequences in a clinical scenario. Various works in recent years have discussed the paradigm of progressive data analytics in detail along with respective potentials and challenges [FFNS19, PMS*21, FP16, TPB*19].

Progressiveness naturally provides approximate uncertain results [HAC*99, FPDs12] that may contain errors [DHC*16] which are potentially corrected at a later step (resulting in a popping artifact for more substantial corrections). Analysts working in progressive scenarios thus need to understand and work effectively under these circumstances. Turkay et al. [TPB*19] conclude that research on progressive approaches needs to consider this uncertainty challenge carefully. Uncertainty in this regard is often communicated via error bars with confidence intervals [HAC*99, FPDs12], and we employ similar means to convey the variance in popping artifacts.

A variety of progressive approaches have been proposed to date, including, among many others, adaptive sampling [ELPZ97, CKK18, PS89, FP04], graph layouting [BP07], and multidimensional scaling [WM04]. Generally, in our work, we consider processes that generate something "spatial", like an image, a volume, or a grid, and in the remainder of this section we outline some of these techniques to demonstrate their breadth and general applicability for many visualization-related tasks. In progressive volume rendering, renderings are increasingly updated by improving the sampling and yielding a more concise representation of the data with fewer errors. In his seminal work, Levoy implemented a volume-rendering algorithm in which image quality is adaptively refined over time [Lev90].

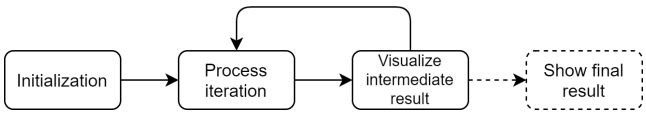

Figure 1: *Iterative refinement process in progressive visualization.*

For evaluation, a visualization of an array shows where rays were cast, where each white pixel corresponds to a single ray. Considering both the spatial domain (the image) as well as the temporal domain (changes over time, e.g., due to user interaction), a method was introduced to dynamically steer the visualization process based on an approximation of respective spatio-temporal errors to achieve interactive frame rates in the visualization of large (time-dependent) volume data sets [FESM14]. Conceptually, a frame is progressively refined by sending additional rays until the temporal error due to changes exceeds the spatial error due to undersampling. A different use case is for time-step selection, which is an iterative optimization process selecting time-steps such that they best cover the full (spatio-temporal) volume data, which allows for an integrated and comprehensive visualization [FE17a]. Supporting selection, progressive approaches have also been used to generate transformations between arbitrary volumes to quickly provide both expressive distances and smooth interpolation [FE17b]. For generating grid layouts, Self-sorting Map uses a permutation procedure to maximize the cross-correlation between member and cell distances by swapping cells in different (sub)quadrants [SG14]. In our research prototype, we use a conceptually similar optimization procedure that also partitions the considered members into sub-groups and locally solves optimization problems.

Ventocilla et al. [VR20] present a model for the progressive visualization and exploration of the structure of large data sets. That is, an abstraction on different components and relations which provide means for constructing a visual representation of a dataset's structure. Visualizations created from corrupt data often mislead users, leading to wrong decisions. Luo et al. [LCQ*20] present a system that progressively visualizes data with improved quality through interactive and visualization-aware data cleaning. Fekete et al. [Fek15] implemented a toolkit which allows analysts to see the progress of their analysis and to steer it while the computation is running.

3. Popping Analysis in Progressive Visualization

Progressive techniques employ iterative methods which improve the result in a step-wise manner and regularly convey a preliminary result to the user (Fig. 1). This process of iterative visualization is what we aim to analyze. For this, we consider a visual outcome to be a 2D spatial object as well as further information associated with it (like metrics or measures from the respective application).

Analysis Questions. In this work, based on own experience as well as our review of related work (as discussed above), we identified four questions that we consider to be relevant:

- How does the refinement generally progress as quantified by expressive metrics and measures?
- When and where do popping artifacts occur across different runs?
- What is the impact of method parameters on popping artifacts?
- What locations are more prone to experience popping artifacts?

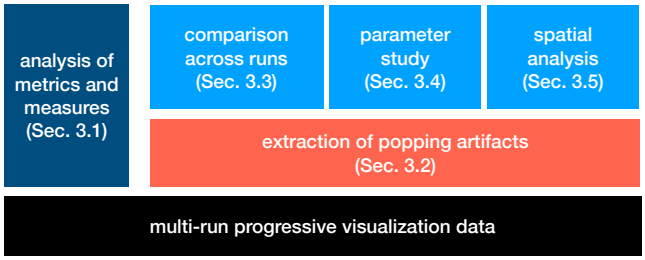

Figure 2: *Framework for the visualization of progressive visualization data (black). Custom metrics and measures regarding provided data can be visualized (Sec. 3.1), but popping artifacts are generally hard to spot in these. For a detailed analysis we explicitly extract popping events (Sec. 3.2), and analyze when and where popping occurred across runs (Sec. 3.3) as well as check the impact of parameters (Sec. 3.4). For scenarios with meaningful absolute positioning, we further analyze popping events in different locations (Sec. 3.5).*

Design Considerations. To address these questions, we introduce a novel visualization framework that provides different perspectives on the runtime characteristics of progressive visualization via different views. For this, we aim to build on simple standard visualization components and concepts as far as possible (like line charts, box plots), while also introducing new custom-tailored visual mappings where necessary. For the development of new designs, we especially consider the special role of spatial positions which constitute the strongest visual cue [Mac86]. The location where popping happens in the visualization space is of importance, but the consideration of parameter spaces and the influence of stochasticity is highly relevant as well. To achieve visual comparison, we employ all three types of comparative visualizations identified by Gleicher et al. [GAW*11]: superimposition (display additional information in the image space), juxtaposition (place visuals side by side to compare across stochastic runs or parameter spaces), and explicit encoding (assess the differences between subsequent iterations). We aim to provide a combination of different views to address our analysis questions in this work. A tighter integration of the proposed components and the integration into an interactive system remains as future work (see Sec. 6). Advanced progressive visualizations are typically adaptive to the data, and with this the concretely observed changes depend on the considered scenario. To get a comprehensive impression of the characteristics of a respective algorithm, different representative examples need to be investigated, and comparisons across them need to be supported (see Sec. 4).

Framework. Our proposed framework consists of dedicated components for yielding new insights regarding progressive visualization characteristics (Fig. 2). Progressive approaches typically have some associated metrics and measures quantifying the progress, like a cost function or difference to the ground truth (Sec. 3.1). While their direct analysis generally conveys an outline of the refinement behavior, the identification of popping artifacts directly from corresponding charts is difficult for various reasons, as exemplified in the use case discussions below. To address this, we particularly aim to provide means for the direct analysis of popping artifacts due to their disturbing nature for simultaneous analysis (see Sec. 3.2 for a discussion on their definition and detection). In our analysis, we consider stochastic refinement schemes which are common in many types of progressive schemes, e.g., to avoid systematic biases in image generation or when using popular randomized search optimization methods (like simulated annealing or evolutionary algorithms). For this, we propose a visualization design that shows the progress of the refinement across different runs (Sec. 3.3). Many approaches typically also depend on parameters, and analyzing their impact regarding popping artifacts is crucial for making an informed choice (Sec. 3.4).

There are some differences between visualization approaches related to the meaning of position. For instance in image generation, the *absolute* position is meaningful as the value of a pixel is specified by the (camera) setup. For such cases, we introduce an overlay to spatially show popping artifacts (Sec. 3.5). In contrast, in other cases only the *relative* position with respect to other objects is important, but not exactly where it is placed in the visualization (e.g., in graph visualization, high-dimensional projection, or tile grids).

3.1. Direct Visualization of Metrics and Measures

Progressive approaches generally come with associated metrics and measures quantifying progress. When considering rendering, image quality metrics give good sense of the state of the refinement (see the use case in Sec. 4). For optimization problems, an objective function that is to be minimized or maximized is a prime example of this (see Sec. 5). Measures provide a good initial impression of refinement behavior and some characteristics thereof, as exemplified in the use cases below. Typically, they are displayed as line charts against time or the number of conducted iterations.

Differences (or the lack thereof) in value can readily be observed in these line charts. In particular, a sudden (between consecutive iterations) significant increase/decrease in the measure can be associated with popping artifacts (assuming the jump is large enough). It can thus give an indication of when to expect popping artifacts. However, in many practical cases popping artifacts are often not clearly visible (e.g., obfuscated by many other parts considered in the metric or not apparent due to overdraw for fine-grained measures).

3.2. Types and Detection of Popping Artifacts

We focus on significant changes between iterations in the visualization results, which we call *popping artifacts*. Popping artifacts are relevant because they introduce disruptive changes to the visualization. A desirable property of progressive/iterative rendering schemes is that changes become continuously less disruptive the longer the process goes on (avoiding popping artifacts later on, if possible). From a developer's perspective, this can be interesting to understand the characteristics of the approach, identify potential for improvement, or to present this as some kind of evaluation. For an end user, this might also be interesting to indicate to what extent further changes might be expected or how stable the results are. In this work, we do not take into account the application context in which the popping occurred or perceptional aspects (see Sec. 6).

Types. We distinguish between three types of popping:

- In *local* popping we consider the change in value of a single element **e** in the visualization.

- In *semi-local* popping we check whether a group of elements **c** have popped for at least G_{pop} elements.
- In *global* popping we consider the change in value of a metric that is measured/computed from the spatial object **O** or by aggregating the individual element values.

Detection. Essentially, we detect popping artifacts when a considered value or metric changes above some threshold T_{pop}. For local or global popping, a popping artifact $\mathcal{P}$ for **L** occurs at iteration n if its metric $\mathtt{M}$ at that iteration has changed more (based on some distance function d) than the specified threshold T_{pop} compared to the previous iteration $n-1$:

$$\mathcal{P}_n(\mathbf{L}) := d(\mathtt{M}_n(\mathbf{L}), \mathtt{M}_{n-1}(\mathbf{L})) > T_{\text{pop}}. \tag{1}$$

The threshold T_{pop} is defined by the user to be meaningful for the respective application context. **L** can be either a single element **e** (local) or the full visualization **O** (global) with $\mathtt{M}$ evaluating the object's value or the aggregate function, respectively. For semi-local popping we apply Eq. 1 for each element in **c**:

$$\mathcal{P}_n(\mathbf{c}) := \left(\sum_{\mathbf{e} \in \mathbf{c}} \mathcal{P}_n(\mathbf{e}) \right) \geq G_{\text{pop}}. \tag{2}$$

Note that Eq. 2 is a generalization of Eq. 1, where we sum over a single object and $G_{\text{pop}} = 1$.

By applying $\mathscr{P}$ at each iteration, we can produce an array of iterations of when popping artifacts occurred:

$$\mathcal{D}(\mathbf{L}) := [n \mid 1 \leq n \leq n_{\max}, \mathcal{P}_n(\mathbf{L}) = \text{True}],$$

where $\mathcal{D}$ is the popping distribution of **L**, which can be either for **e** (local), **c** (semi-local), or **O** (global). These distributions can neatly be visualized by boxplots. All in all, this establishes *when* and *where* popping artifacts occur.

3.3. Multi-Run Overview Visualization of Popping Artifacts

In general, the results of a single stochastic run are not too meaningful for the analysis, and many runs need to be considered to get a stable overall impression. We propose a generic visualization design for popping artifacts across multiple runs (Fig. 3). In this visualization, multiple runs are stacked on top of each other for comparison. The visualization consists of the following components.

The **metric bar** is a scalar value measured/computed from the intermediate results and is specified by the user. It is visualized as a color bar by mapping the metric to color using a color map, and in order to achieve this the values have to be normalized. Because we consider several different runs, the normalization has to be applied to the matrix (as opposed to each run individually) so that the same color corresponds to the same metric value. While the changes between consecutive iterations may be difficult to see, it still gives a general impression of the changes in the metric. The advantage over line plots is that they scale much better to a larger number of runs.

The **triangles** indicate iterations at which global/semi-local/local popping artifacts occurred, which is essentially the distribution $\mathcal{D}$.

The **boxes of spatial changes** (a.k.a. blocks) show where significant changes occurred and with this caused the popping. If the user decides to visualize local popping, all the individual elements that

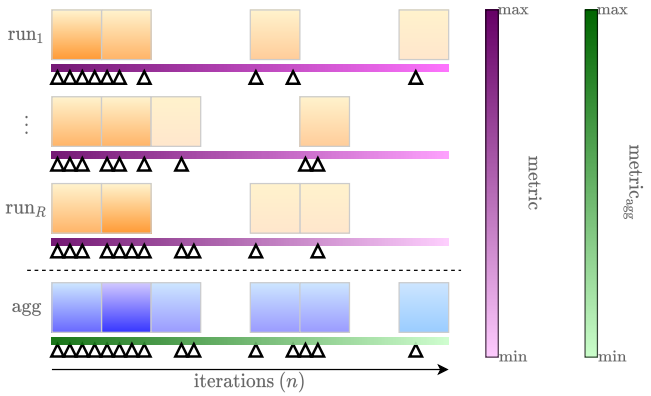

Figure 3: *Multi-run visualization for popping artifacts of R different runs. Small images (orange) highlight spatial changes after popping artifacts (triangles). The metric bar (purple) shows the metric value at all steps. The runs are reordered such that similar ones are next to each other. The last row aggregates all aspects of the runs.*

popped are highlighted. In the case of global popping all changes to the visualization are shown. In any case, it shows when and where popping occurred. As space is limited, we can only display some number of blocks for each run. To still convey spatial popping in a way that is comparable across runs, we divide the iterations into intervals of size b and aggregate popping artifacts within. Our primary focus in this component is to show how often popping artifacts occurred in certain places. To convey this, we count all popping artifacts at each position and finally normalize within each box with respect to the maximum. This gives values in $[0,1]$, which are color mapped. Note that this means that the same color between different blocks does not necessarily correspond to the same frequency, but it yields a much clearer impression for each block individually.

The popping of a single element might be difficult to see, especially for higher resolution structures, unless zoomed in. Therefore, the elements indicating popping are enlarged by taking the maximum value of a certain radius for each element.

Reordering the runs improves the visual clarity and allows for better comparability if looking at larger numbers of runs. They are reordered using hierarchical clustering based on the iterations at which popping occurred (represented by a binary vector). A distance function computes the similarity between two popping vectors of different runs, such the cosine or Hamming distance.

In some scenarios developers might choose to consider a significant number of runs to get a comprehensive overall impression. While the visual representation discussed so far could be extended in a straight-forward way, this would lead to issues regarding visual scalability. To address this, we include another row which **aggregates** all the runs to summarize them, e.g. by averaging or taking the median. To distinguish the aggregate row from the runs we apply a different color map (although the same one could be used as well because each block is normalized locally). Similarly, aggregates are also computed for metrics, and we add a dedicated metric bar as the value range can differ (e.g., when considering variance).Then, aggregated popping is computed by aggregating all popping artifacts

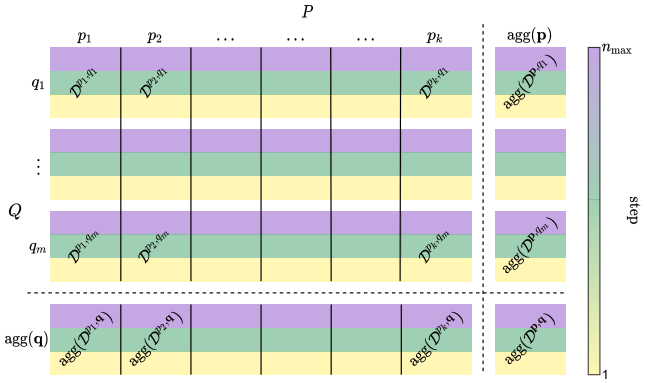

Figure 4: *Parameter space analysis for parameters P and Q, with k and m different values, respectively. The bottom row (resp. right column) aggregates the distributions of the respective columns* ***q*** *(resp. rows* ***p****), and the bottom right aggregates all the distributions.*

for each step across runs. Depending on the concrete question, we can, e.g., take the union of all the popping artifacts to indicate that in at least one run popping happened, or take the intersection to indicate that popping artifacts occurred in all runs.

This visualization is flexible in the sense that the user can specify two metrics: one for the color bar and one for the popping; and they can choose between local and global popping.

3.4. Parameter Space Analysis of Popping Artifacts

Progressive visualizations have parameters which can attain a range of different values. Naturally, the choice of the parameter values influences the popping behaviour as they change the output of the intermediate results. Besides the algorithm parameters, the popping analysis requires the user to set the popping threshold T_{pop} (and G_{pop} for semi-local popping). Generally, there are many parameters to consider, but some have more influence than others.

To visualize the impact of the parameter space on the popping behaviour, we visualize the popping distribution per parameter combination for several values each (Fig. 4). That is, we create a rectangular grid whose dimensions depend on the number of specified parameter values, and within the cells for each parameter combination show a box plot depicting the occurrence of popping artifacts. An individual 2D grid can visualize the impact of two parameters (this could easily be extended to more parameters akin to scatterplot matrices [EGS*13]). Besides the box plots, three colour bands are vertically stacked in the cells' background to provide additional visual cues: the bottom one corresponds to the minimum popping iteration, the middle color to the median popping iteration, and the top color to the maximum popping iteration. This allows for easier comparison between box plots on different rows.

Including a large number of parameter values could lead to an explosion of distributions, complicating the analysis of a single parameter combination. Therefore, an additional row/column aggregates the distributions of the respective columns/rows to summarize them. Moreover, the bottom-right block aggregates all distributions.

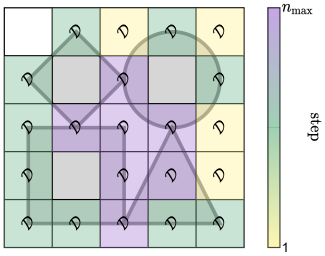

Figure 5: *Structure (represented by shapes) overlaid with a* 5×5 *uniform grid, where each cell shows the popping distribution* $\mathcal{D}$ *and a colored background according to its median popping time step.*

3.5. Spatial Popping Artifacts

Popping artifacts are caused by the spatial objects, which have a position in the structure. Spatial objects with an absolute position are always in the same location, as opposed to ones with a relative position. That is, for the former, if a popping artifact occurred at the same location between two different runs, we know the same spatial object popped, which is not the case for the latter. Hence, we overlay the structure with cells and box plots over the spatial objects to spatially show their popping distributions (Fig. 5). Moreover, the grid cells' backgrounds are colored with respect to their median time step. Combining this with a grey-scale version of the structure (if applicable) allows for easier interpretation of the popping. Cells where no popping occurred do not have a color or show a distribution. This shows the user *where* the popping artifacts occurred in addition to when. Note that, while one could apply this visualization to the latter case too, this would in general not be meaningful.

In our implementation, the structure is overlaid with a uniform rectangular grid that precisely fits it. This visualization could be extended to work with non-uniform rectangular grids or other grid shapes (e.g. hex-grids) for that matter. Further, adaptive grids could be employed with smaller grid cells in regions of interest and larger ones for less interesting areas. When combining multiple runs, the naive approach would be to create the popping distribution image for all of them and compare them as is. However, the presented distributions can also be combined directly to allow to scale to a larger number of runs. An alternative could be to compute some metric for this comparison (i.e., explicit encoding [GAW*11]). There is a trade-off between the scaling/visibility of the boxplots (related to the cell size) and popping localization. Increasing the dimensions of the grid cells improves the visibility of the boxplots, but larger grid cells capture popping at a coarser granularity over a greater area, impeding the precise localization of popping artifacts.

4. Adaptive Sampling in Image Space

The adaptive image space sampling use case simulates a ray-tracing application with Monte Carlo-based sampling (see Zwicker et al. [ZJL*15] for a comprehensive overview). Starting from a coarse uniform sampling, a certain number of samples is added in each iteration (we use 1% of the number pixels, i.e., 655 for 256×256 images). A full image can be reconstructed at each iteration via interpolation (Fig. 6). Sampling locations are determined in a stochastic manner based on the color variation of the pixels. The process completes when all pixels have been sampled (after 100 iterations). Pixels are considered to be individual elements **e** and the (reconstructed) images are spatial objects **O**.

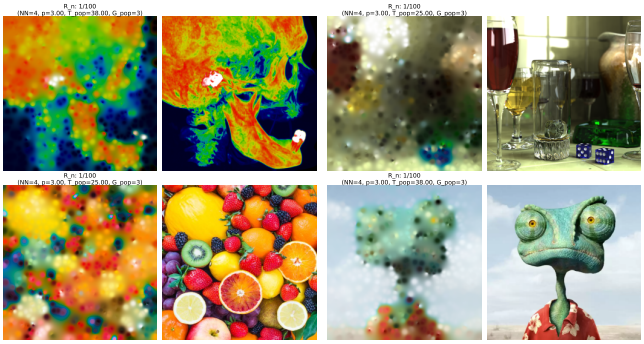

Figure 6: *Image pairs of the initial and final reconstruction (top left to bottom right: maximum intensity projection of a human head [Ros21]; photo-realistic scene with translucent and reflective objects created by a ray-tracing application; photo of some fruit; a still frame of the 2011 computer-animated film Rango [Ver11].)*

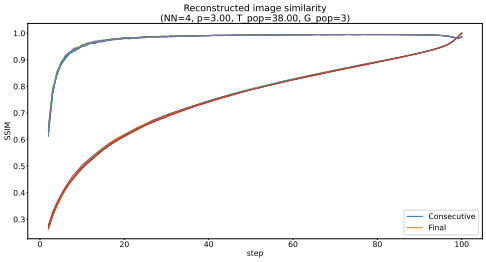

Figure 7: *Structural similarity of reconstructed images against their predecessor (consecutive) and the final image (of the head).*

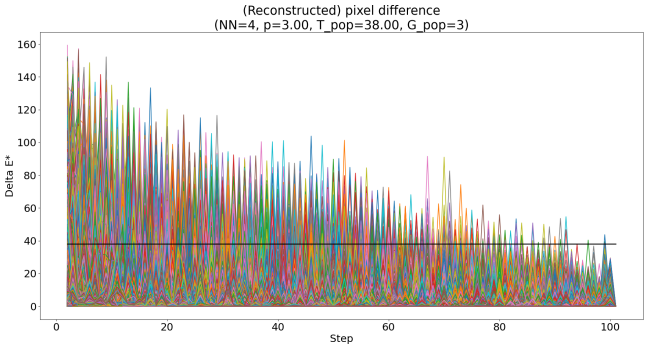

Figure 8: *Plot of the color distance between two consecutive reconstructed images for every pixel for the head input image.*

4.1. Refinement Characteristics

To get an initial impression on the refinement behavior, we compute the similarity between two consecutive reconstructed images: $\mathrm{sim}(R_n, R_{n-1})$, where $\mathrm{sim}(\cdot,\cdot)$ could be any similarity metric. This is a global metric of the reconstructed image and takes into account the values of all individual pixels. We further compute $\mathrm{sim}(R_n, I)$ to compare the intermediate results with the final result. We can easily combine the similarity graphs for several runs into a single figure, which shows quite similar refinement characteristics overall.

Image differences (Fig. 7). We first assess the progress globally via the structural similarity index (SSIM) [WB09], which considers the perceived change in structural information. The first graph for comparing consecutive reconstructed images shows a sharp increase for the first few iterations. This is because the images are reconstructed from a relatively small number of pixels, and with each new sample we can reconstruct the images considerably better. After that, it stabilizes (with small fluctuations) because the images remain similar as the new samples do not add much new information. Near the end, however, there is a small drop, indicating a change in structure. This is caused by the constant regions in the image, such as the teeth. They are the last to be sampled, meaning that up till then, they are reconstructed from pixels far away, and the interpolated color does not have to be close to their true color. Once they are sampled, their reconstruction could thus cause a change in structure large enough to cause the similarity to drop.

The second graph for comparing reconstructed images with the final image is different. It also increases, but not as sharply. Unlike the first graph, the second one does not experience the sudden change in structure, because it is not revealed during the iterations, but already known beforehand. This is also why the similarity is predominantly lower for the second graph. Near the end, the rapid increase in similarity indicates that the adaptive sampling scheme could be improved (either by tuning parameter values or employing a totally different sampling scheme). Afterwards, the image is perfectly "reconstructed" with a structural similarity index of 1.

(Reconstructed) pixel differences (Fig. 8). While the graphs discussed above demonstrate general refinement characteristics, popping artifacts cannot be identified individually as they get averaged out in the global view. To address this, we compute the color distance between the reconstructed pixels, i.e., the Euclidean distance between the color channels, at time step n and the previous time step $n-1$: $d(R_n(\mathbf{p}), R_{n-1}(\mathbf{p}))$. We then plot these distances versus the iterations, which gives an idea of how the pixel colors change over time. We add a horizontal line at T_{pop} in the signal plot to show that all peaks above this line are considered local popping artifacts.

First of all, we observe that the overall distance decreases with each iteration, meaning that consecutive reconstructed images become more alike. Moreover, many pixels spike each iteration. The black horizontal line shows the popping threshold, and so all spikes above this line are considered popping artifacts. The lines continue to spike even near the final iterations, but below the threshold, and so they are not considered as popping artifacts.

In general, we see many spikes: the lines go from low values to high values, and back to low values. However, in the first few iterations some of the lines stay at high values. This means that drastic changes happen and pixels pop several times. These changes in the first few iterations can also be observed in Fig. 7 for the similarity between consecutive images. However, this visualization quickly becomes cluttered because of the large number of pixels.

4.2. Popping detection

We now explicitly detect popping artifacts, addressing the above issues of them being either averaged out or resulting in serious overdraw. We are interested in groups of popped pixels. Hence, we

Figure 9: *Image overlaid with a* 5×5 *uniform rectangular grid. The gray cell under consideration is expanded by* 50% *in all directions for popping detection.*

utilize Eq. 2 and replace the spatial object **e** with pixel **p** and the metric `M` is the pixel (reconstructed) color value. The regions **c** are created by overlaying the image with a uniform rectangular grid with cells. Then, we check for popping artifacts inside those grid cells. The popping detection is based on the sum of changes within a region, but using the grid cells directly might lead to results that are heavily dependent on the positioning of the grid (i.e., unfortunate positioning could split an area of substantial change into four parts and it might not be detected). To address this, the grid cells are uniformly expanded by 50% so that we cover larger areas and every part of the image is captured twice; see Fig. 9. Semi-local popping for this use case is thus defined as a threshold for the change of color value for a group of pixels.

In our experiments, we identified meaningful parameters for popping detection as follows: the popping threshold is set to 38 and the popping group size to 3. The grid resolution is 16×16, so every grid cell is 16×16.

4.3. Multi-Run Overview Visualization of Popping Artifacts

For our study, we perform 8 different runs of 100 iterations (Fig. 10). For the color bar, we choose the similarity to the final image (as shown in Fig. 7). The small images show individual pixels that popped, i.e., local popping, while the triangles represent the semi-local popping of groups of pixels. That way, missing triangles indicate that no region had a sufficient number of popping, but the small images might still show that locally pixels changed significantly.

We see that for all runs many popping artifacts occurred in the first ≈ 60 iterations, after which the popping rate slows down. We observe the same behavior in the small images, where especially in the first block many pixels popped. For later iterations, we see blocks which faintly show local popping but not enough for the semi-local popping, as indicated by the missing triangles. All the runs show similar results in terms of where and when popping happened, indicating that the process is stable after about 60 iterations. However, it can be seen that with the considered adaptive sampling strategy in some cases popping artifacts can still occur very close toward the end with only few pixels left to sample. The popped pixels are enlarged by taking the maximum in 5 pixel radius.

In addition, Fig. 10 shows the aggregated rows for the other three input images (see the supplementary material). We observe similarities (all popping continues to occur around edges and the head/fruit-rows pop until the final iterations) and some differences (the Rango-row stops early with popping, and the glasses-row pops again after a period of no popping).

4.4. Parameter Study

We now investigate the impact of two crucial parameters (Fig. 11). First, the *number of nearest neighbors* (in $\{2,4,8\}$ with 4 being the

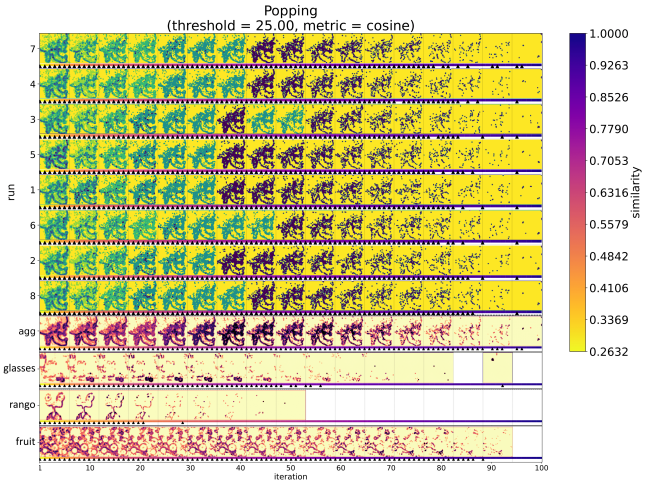

Figure 10: *Multi-run comparison of the adaptive image space sampler. It is applied to* 8 *different runs of* 100 *iterations with interval size* $b = 6$ *using the cosine distance for reordering, with* $T_{\text{pop}} = 25$. *The last row aggregates the small images and metrics by averaging, and the popping by taking the union. Moreover, the aggregated rows of the other three input images are included*

default) is used in the sampling scheme and indicates how many nearest neighbors are to be considered for determining the color variation and deciding which pixels to sample next. Second, the *power parameter* (in $\{1.5,3,6,12\}$ with 3 as the default) is used in the reconstruction scheme in the inverse distance weighting method for multivariate interpolation, where it controls how much the distance (inversely) affects the weight. Higher values assign greater influence to values closest to the interpolated point, while lower values cause the interpolated values to be dominated by points far away. For each parameter combination 4 different runs are performed.

Overall, we can observe that the power parameter has the most substantial impact. For power parameter 1.5, popping artifacts occurred for all time steps, and afterwards get increasingly smaller for larger values. The number of neighbors also has a significant impact, however, not as prominent. The parameter combination with the least amount of popping appears to be for number of nearest neighbors 2 and power parameter 12.

With respect to the adaptive sampling method, this indicates that popping behaviour depends more on the reconstruction scheme (influenced by the power parameter) than the sampling scheme (influenced by the number of nearest neighbours). Nevertheless, one would ideally include more parameter values and runs for a more detailed parameter study.

4.5. Positional Analysis: Popping & Sampling Time

We now investigate where in an image popping tends to happen.

Popping distribution. As described in Sec. 4.2, we look for semi-local popping of the pixel colors in expanded grid cells. Fig. 12 shows the popping distribution image with a gray-scale version of R_n to clearly show where and when the popping occurred. As expected, a lot of popping happened in the early stages of the process,

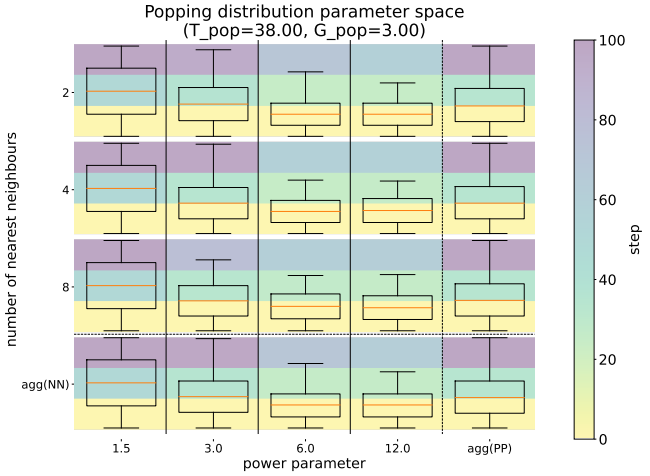

Figure 11: *2D parameter space analysis of adaptive image space sampling for the number of nearest neighbors and power parameter for the head input image. The aggregation is done through merging.*

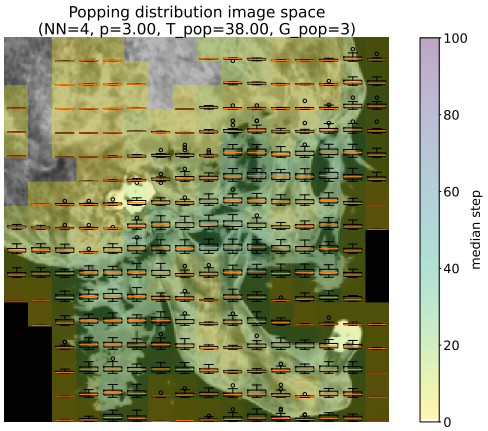

Figure 12: *Temporal distribution of popping artifacts on a* 16×16 *grid (*$T_{\text{pop}} = 38$*,* $G_{\text{pop}} = 3$*).*

as indicated by the yellow colors. For some of those, the popping happened *only* at the the early stages, as indicated by the thin boxplots. This mainly happens in smoother and more constant regions.

Judging from the green colors, most popping artifacts occurred in the neck and middle part of the skull where there is a lot of color variation. Moreover, the wider boxplots indicate that they happened at many time steps, from the early stages until the late stages. No popping happened in the background regions.

Sampling. For a close analysis, we check when each pixel **p** was sampled. While it does not visualize the popping itself, it shows the sampling behaviour of the iterative process which can be used to infer this information. In this image, the time step is mapped to a color so that we can see when it was sampled. Moreover, to give a better understanding of why those pixels were sampled, this image is overlaid on a gray-scale version of the reconstructed image (as to not interfere with the colors of the sampling).

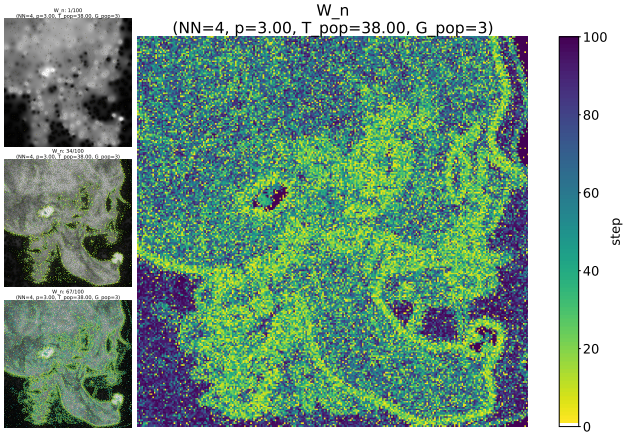

Figure 13: *Left column: When pixels were sampled with gray-scale version of reconstructed image for* W_1*,* W_{34} *and* W_{67}*. Right: Final visualization* W_{100} *showing when pixels are sampled with a color bar (without gray-scale).*

Fig. 13 shows when and which pixels were sampled on top of a gray scale image of the intermediate results. The randomness in the sampling is clearly visible, but we also find the structures of the image. Note how the sampling favors edges because of their sharp transitions, i.e., high color variation, meaning they are sampled first. The sampling then continues on other parts of the skull, which contain smoother color transitions. Occasionally we see pixels being sampled in the background regions, but they are sampled last for the most part. In the intermediate results (Fig. 6) we see that the teeth are only reconstructed correctly near the final iteration. We infer from this visualization that that is because they are sampled very late, and so up till then their reconstruction is off.

5. Tile Grid Generation

In the second use case, we consider a current research prototype for the placement of images in a grid such that they are as similar as possible to their neighbors [SG14, FDH*15, QSST10]. In each of 1024 considered iterations, images are randomly partitioned into groups of size k, maximizing neighborhood similarity by swapping tiles within. We aim to identify when and how often larger changes occur, how stable this is across runs, and what the impact of group size k is on this behavior. This can further indicate when a run can be deemed to be sufficiently stable, as there is no inherent criterion for convergence of the procedure. Note that in this use case, absolute positions are not meaningful—only relative positioning between tiles is—and accordingly we omit location-based analysis here.

As exemplified in Fig. 14, generally a large number of changes induces popping artifacts early on ($\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$), while later iterations appear more stable with fewer popping ($\mathbf{G}_{64}, \mathbf{G}_{128}, \mathbf{G}_{1024}$).

5.1. Refinement Characteristics

Cost progression resembles exponential decay, which is caused by the random initialization and the nature of stochastic processes (Fig. 15). It is clear that popping artifacts occur in the first

Figure 14: *Intermediate results of grid layout generation for 1024 images from the Caltech 101 database [LAR03] (from left to right):* $\boldsymbol{G}_1$ *(initialization),* $\boldsymbol{G}_2, \boldsymbol{G}_3, \boldsymbol{G}_{64}, \boldsymbol{G}_{128}$ *and* $\boldsymbol{G}_{1024}$ *("final" grid).*

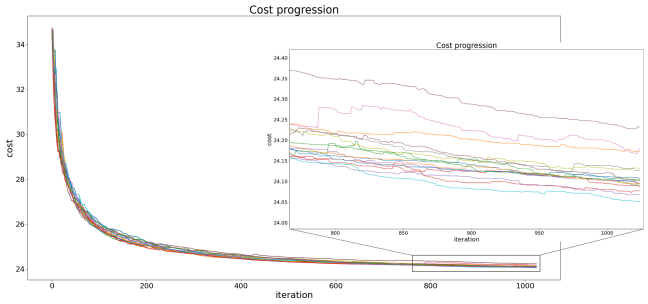

Figure 15: *Cost of the tile grid generation process versus the iteration, imposed with a zoom in on the final* 256 *iterations.*

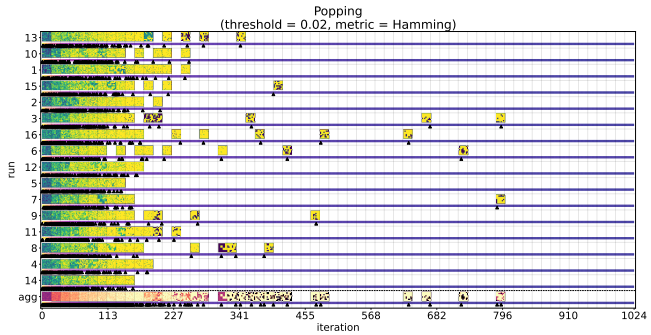

Figure 16: *Multi-run visualization for the tile placement use case featuring* 16 *different runs of* 1024 *iterations, each with interval size* $b = 16$ *(the Hamming distance is used for reordering). The last row aggregates the small images and metrics by averaging, and the popping by taking the union.*

iterations (in the drop). However, this is not obvious later on where the graph shows little change. Once zoomed in, it is clear that the graph continues to fluctuate, albeit in much smaller magnitude.

5.2. Popping Detection

In the optimization the cost is minimized and is thus an important metric to consider. (Global) popping for this use case is defined as a threshold regarding changes of cost. If the user has a priory knowledge about the cost values, they can readily set the threshold. Otherwise, it can be determined by examining the direct cost (Fig. 15) in connection with the actual grids (Fig. 14). As the changes are small later on, the popping threshold is small as well, namely $T_{\text{pop}} = 0.02$.

5.3. Multi-Run Overview Visualization of Popping Artifacts

The multi-run overview visualization used the following parameter settings for this use case: the interval size $b = 16$ and the hierarchical cluster reordering uses the Hamming distance. The visualization for the popping artifacts across multiple runs is shown in Fig. 16. No enlargement of individual elements is used. We see that for all the runs many popping artifacts occurred in the first 113 iterations. As expected, later on the popping is less frequent. However, for several runs $(3, 6, 7)$ popping happened near the end, yielding disruptive changes late in the refinement process. More notably, some runs stop popping (in the given number of iterations, that is) relatively early (after ≈ 256 iterations), while others sporadically pop later on.

Especially in the first few small images, the majority of the tiles have swapped positions. The fact that the tile swapping happens in groups is reflected by the apparent squares. That is, when two such squares are swapped and caused a large enough change in cost, all tiles in those squares contributed to that popping and are highlighted in the small image. Because this use case works with relative positions, the small images show no underlying structure, as opposed to Fig. 10.

5.4. Parameter study

For the grid layout generation we study the effects on the popping behavior of parameter k. We further consider the impact of the popping threshold parameter T_{pop}, which defines what degree of popping we observe in the visualization. Note that these are two different things that we are varying (e.g. in Fig. 11, both parameters tweak adaptive sampling; for tile placement, only one is a method parameter and the other one defines what degree of popping we observe in the visualization).

Fig. 17 shows the parameter space visualization for $k \in \{2, 5, 10, 20, 40\}$, $T_{\text{pop}} \in \{0.005, 0.01, 0.02\}$. Overall, it can be seen that larger k leads to much fewer changes later on, results in more favorable progression characteristics, and yields lower cost values, but requires more computation time (see the supplementary material).

For all parameter values popping occurred in the first step(s). More importantly, for all three values of T_{pop} (but mainly for 0.005) we observe that the median popping time step decreases as k increases in a qualitatively similar fashion, providing an important indication regarding the stability and expressiveness of the visualization results. This generally holds for the maximum popping time

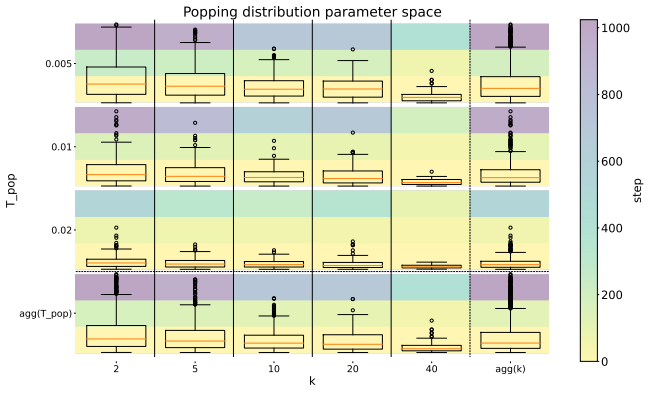

Figure 17: *Parameter study of grid layout generation with group size k and popping threshold T_{pop} (with aggregation via merging.)*

step as well, although there are some changes of interest. In general, considering different T_{pop} also provides an impression of the degree of significance of occurring popping events.

6. Discussion and Conclusion

To the best of our knowledge, we have proposed the first dedicated framework for analyzing popping artifacts—significant changes in the visualization from one iteration to another—in progressive visualization, a paradigm that has substantially gained in popularity in recent years due to growing sizes of datasets and increasing complexity of advanced methods. While our framework also considers general refinement characteristics, we focus on popping artifacts due to their high relevance for simultaneous user analysis. Progressive visualization is employed in diverse settings, and the goal of this work was to propose a framework that can adequately capture commonalities across them while being flexible enough to account for conceptual differences.

To exemplify this, we applied our framework (Sec. 3) to two substantially different use cases: adaptive image space sampling in Sec. 4 and the generation of grid layouts in Sec. 5. While they share general similarities of progressive visualization approaches, they also have some fundamental differences. For example, in the adaptive image space sampler, once a pixel is sampled at a certain location in the image it stays fixed. In contrast, the tiles have no final position and can change at any time step in the tile grid generation.

In our framework, this is represented by the fact that some visualizations are specific to a class of use cases (e.g, for spatial analysis or parameter studies), while we considers others to be generic, such as our multi-run overview visualization. For example, the generic visualization would also work on a third use case doing graph processing or could be extended to approaches working on 3D datasets. As motivated and discussed in detail in Sec. 3, the different components of our framework have been designed to provide a complementary view on different perspectives. For example, regarding the adaptive sampling we see that it yields similar progression despite the stochasticity involved, with popping artifacts occurring in regions of high color variation all the way to the end when the final pixels are sampled. From the parameter analysis we learned that the power parameter has a much more significant influence on the popping than the number of neighbors (with higher power values being beneficial). For the tile grid generation, runs are much more heterogeneous in comparison, with some runs exhibiting quite smooth refinement from early on while another random initialization can result in (several) later popping artifacts. The parameter study showed that larger group sizes lead to fewer changes later on, and varying the popping threshold shows both the occurrence of different degrees of popping and crucially the qualitative stability of results.

Our presented framework currently consists of individual components providing different perspectives. In future work, we aim to integrate the visualizations into a generic and flexible interactive visual analytics tool for the analysis of progressive visualization. In particular, this would involve adding interaction modalities and linking different views and components. We further aim to consider further diverse use cases—like graph visualization or high-dimensional data analysis—and on this basis determine further perspectives to integrate into our framework. With the prospective visual analytics tool we plan to conduct user and real-world evaluations. In particular, we aim to assess the added utility of different components in comparison to standard alternatives, and collect feedback to further improve our approach. In doing this, we also aim to consider additional use cases from the workflow of (expert) users to yield a representative set overall. This supplements the two currently considered use cases which were chosen to demonstrate the flexibility of the framework while also being closely aligned with a commonly used techniques (adaptive sampling) as well as a current research prototype (tile placement).

How impactful popping artifacts are in general depends on the application context, which we plan to consider explicitly in future work. For example, there is a qualitative difference between popping in near-real-time renderers that use amortization/approximation techniques to reduce latency for user interaction and the acceleration of non-real-time solutions to reduce time of delivering of (first) intermediate solutions. We further aim to explicitly account for human perceptional aspects regarding popping artifacts and how these affect the user experience, e.g., by distinguishing between focus and context regions as also implemented in foveated rendering approaches [GFD*12, BSB*19, FBB*21]. We also plan to consider advanced methods for mitigating popping effects, like smooth transitions or fading between levels of detail. While they reduce the impact of popping, they also delay the improvement of the presented results with results from the latest iteration. Accordingly, this could be reflected in the popping formulations (Eq. 1 & Eq. 2) and visualizations as a transition instead of a singular event.

Regarding visual scalability, our multi-run visualization proposed in this work has been demonstrated with up to 16 runs, with added aggregation for a clear summary. We already reorder runs based on similarity, and we could further combine the results and show the most representative ones. For large parameter studies, the aggregation of subsets of parameter value clusters could create larger blocks and improve visibility. In addition, more than two parameter dimensions could be considered and presented in a small multiples layout [Tuf90, WBWK00].

Acknowledgment This paper is in part based on the first author's MSc research internship at the University of Groningen.

References

[BP07] BRANDES U., PICH C.: Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing* (Berlin, Heidelberg, 2007), Kaufmann M., Wagner D., (Eds.), Springer Berlin Heidelberg, pp. 42–53. doi:10.1007/978-3-540-70904-6_6. 2

[BSB*19] BRUDER V., SCHULZ C., BAUER R., FREY S., WEISKOPF D., ERTL T.: *Voronoi-Based Foveated Volume Rendering*. The Eurographics Association, 2019. doi:10.2312/evs.20191172. 10

[CKK18] CHRISTENSEN P., KENSLER A., KILPATRICK C.: Progressive Multi-Jittered Sample Sequences. *Computer Graphics Forum 37*, 4 (2018), 21–33. doi:10.1111/cgf.13472. 2

[DHC*16] DING B., HUANG S., CHAUDHURI S., CHAKRABARTI K., WANG C.: Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. In *Proceedings of the 2016 International Conference on Management of Data* (New York, NY, USA, June 2016), SIGMOD '16, Association for Computing Machinery, pp. 679–694. doi:10.1145/2882903.2915249. 2

[EGS*13] EMERSON J. W., GREEN W. A., SCHLOERKE B., CROWLEY J., COOK D., HOFMANN H., WICKHAM H.: The Generalized Pairs Plot. *Journal of Computational and Graphical Statistics 22*, 1 (Jan. 2013), 79–91. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2012.694762. doi:10.1080/10618600.2012.694762. 5

[ELPZ97] ELDAR Y., LINDENBAUM M., PORAT M., ZEEVI Y.: The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing 6*, 9 (Sept. 1997), 1305–1315. Conference Name: IEEE Transactions on Image Processing. doi:10.1109/83.623193. 2

[FBB*21] FRIESS F., BRAUN M., BRUDER V., FREY S., REINA G., ERTL T.: Foveated Encoding for Large High-Resolution Displays. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (Feb. 2021), 1850–1859. Conference Name: IEEE Transactions on Visualization and Computer Graphics. doi:10.1109/TVCG.2020.3030445. 10

[FDH*15] FRIED O., DIVERDI S., HALBER M., SIZIKOVA E., FINKELSTEIN A.: Isomatch: Creating informative grid layouts. *Computer Graphics Forum 34*, 2 (2015), 155–166. doi:10.1111/cgf.12549. 8

[FDK12] FISHER D., DRUCKER S. M., KÖNIG A. C.: Exploratory Visualization Involving Incremental, Approximate Database Queries and Uncertainty. *IEEE Computer Graphics and Applications 32*, 4 (July 2012), 55–62. Conference Name: IEEE Computer Graphics and Applications. doi:10.1109/MCG.2012.48. 2

[FE17a] FREY S., ERTL T.: Flow-based temporal selection for interactive volume visualization. *Computer Graphics forum 36* (2017), 153–165. doi:10.1111/cgf.13070. 2

[FE17b] FREY S., ERTL T.: Progressive direct volume-to-volume transformation. *IEEE Transactions on Visualization and Computer Graphics 23* (2017), 921–930. doi:10.1109/TVCG.2016.2599042. 2

[Fek15] FEKETE J.-D.: ProgressiVis: a Toolkit for Steerable Progressive Analytics and Visualization. In *1st Workshop on Data Systems for Interactive Analysis* (Chicago, United States, Oct. 2015), p. 5. URL: https://hal.inria.fr/hal-01202901. 2

[FESM14] FREY S., ERTL T., SADLO F., MA K.: Interactive progressive visualization with space-time error control. *IEEE Transactions on Visualization and Computer Graphics 20* (2014), 2397–2406. doi:10.1109/TVCG.2014.2346319. 2

[FFNS19] FEKETE J.-D., FISHER D., NANDI A., SEDLMAIR M.: Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411). *Dagstuhl Reports 8*, 10 (2019), 1–40. Place: Dagstuhl, Germany Publisher: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/DagRep.8.10.1. 1, 2

[FP04] FARRUGIA J.-P., PÉROCHE B.: A Progressive Rendering Algorithm Using an Adaptive Perceptually Based Image Metric. *Computer Graphics Forum 23*, 3 (2004), 605–614. doi:10.1111/j.1467-8659.2004.00792.x. 2

[FP16] FEKETE J.-D., PRIMET R.: Progressive Analytics: A Computation Paradigm for Exploratory Data Analysis. *arXiv:1607.05162 [cs]* (July 2016). arXiv: 1607.05162. URL: http://arxiv.org/abs/1607.05162. 1, 2

[FPDs12] FISHER D., POPOV I., DRUCKER S., SCHRAEFEL M.: Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, May 2012, pp. 1673–1682. doi:10.1145/2207676.2208294. 2

[GAW*11] GLEICHER M., ALBERS D., WALKER R., JUSUFI I., HANSEN C. D., ROBERTS J. C.: Visual comparison for information visualization. *Information Visualization 10*, 4 (Oct. 2011), 289–309. doi:10.1177/1473871611416549. 3, 5

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3D graphics. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 164:1–164:10. doi:10.1145/2366145.2366183. 10

[HAC*99] HELLERSTEIN J., AVNUR R., CHOU A., HIDBER C., OLSTON C., RAMAN V., ROTH T., HAAS P.: Interactive data analysis: the Control project. *Computer 32*, 8 (Aug. 1999), 51–59. Conference Name: Computer. doi:10.1109/2.781635. 2

[Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings Visualization '98 (Cat. No.98CB36276)* (1998), pp. 35–42. doi:10.1109/VISUAL.1998.745282. 1

[LAR03] LI F.-F., ANDREETTO M., , RANZATO M. A.: Caltech 101, 2003. URL: http://www.vision.caltech.edu/Image_Datasets/Caltech101/. 9

[LCQ*20] LUO Y., CHAI C., QIN X., TANG N., LI G.: Visclean: Interactive cleaning for progressive visualization. *Proc. VLDB Endow. 13*, 12 (Aug. 2020), 2821–2824. doi:10.14778/3415478.3415484. 2

[Lev90] LEVOY M.: Volume rendering by adaptive refinement. *The Visual Computer: International Journal of Computer Graphics 6* (1990), 2–7. doi:10.1007/BF01902624. 2

[Mac86] MACKINLAY J.: Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics 5*, 2 (Apr. 1986), 110–141. doi:10.1145/22949.22950. 3

[PMS*21] PROCOPIO M., MOSCA A., SCHEIDEGGER C. E., WU E., CHANG R.: Impact of cognitive biases on progressive visualization. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1. Conference Name: IEEE Transactions on Visualization and Computer Graphics. doi:10.1109/TVCG.2021.3051013. 1, 2

[PS89] PAINTER J., SLOAN K.: Antialiased ray tracing by adaptive progressive refinement. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, July 1989), SIGGRAPH '89, Association for Computing Machinery, pp. 281–288. doi:10.1145/74333.74362. 2

[QSST10] QUADRIANTO N., SMOLA A. J., SONG L., TUYTELAARS T.: Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 10 (Oct 2010), 1809–1821. doi:10.1109/TPAMI.2009.184. 8

[Ros21] ROSSI R. A.: Volume rendering, maximum intensity projection and isosurfaces, 2021. URL: http://ryanrossi.com/sv3.php. 6

[RS09] ROSENBAUM R., SCHUMANN H.: Progressive refinement: more than a means to overcome limited bandwidth. In *Visualization and Data Analysis 2009* (Jan. 2009), vol. 7243, SPIE, pp. 145–156. doi:10.1117/12.810501. 2

[SG14] STRONG G., GONG M.: Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. *IEEE Transactions on Multimedia 16*, 4 (June 2014), 1045–1058. doi:10.1109/TMM.2014.2306183. 2, 8

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE*

Transactions on Visualization and Computer Graphics 20, 12 (Dec. 2014), 1653–1662. Conference Name: IEEE Transactions on Visualization and Computer Graphics. doi:10.1109/TVCG.2014.2346574. 2

[SS09] SCHWARZ M., STAMMINGER M.: On predicting visual popping in dynamic scenes. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization* (New York, NY, USA, 2009), APGV '09, Association for Computing Machinery, p. 93–100. doi:10.1145/1620993.1621012. 1

[TPB*19] TURKAY C., PEZZOTTI N., BINNIG C., STROBELT H., HAMMER B., KEIM D. A., FEKETE J.-D., PALPANAS T., WANG Y., RUSU F.: Progressive data science: Potential and challenges, 2019. arXiv:1812.08032. 1, 2

[Tuf90] TUFTE E. R.: *Envisioning Information*. Graphics Press, 1990. 10

[Ver11] VERBINSKI G.: Rango, 2011. Feature film. 6

[VR20] VENTOCILLA E., RIVEIRO M.: A model for the progressive visualization of multidimensional data structure. In *Computer Vision, Imaging and Computer Graphics Theory and Applications* (Cham, 2020), Cláudio A. P., Bouatouch K., Chessa M., Paljic A., Kerren A., Hurter C., Tremeau A., Farinella G. M., (Eds.), Springer International Publishing, pp. 203–226. doi:10.1007/978-3-030-41590-7_9. 2

[WB09] WANG Z., BOVIK A.: Mean squared error: Love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine 26* (2009), 98–117. doi:10.1109/MSP.2008.930649. 6

[WBWK00] WANG BALDONADO M. Q., WOODRUFF A., KUCHINSKY A.: Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on Advanced visual interfaces* (New York, NY, USA, May 2000), AVI '00, Association for Computing Machinery, pp. 110–119. doi:10.1145/345513.345271. 10

[WM04] WILLIAMS M., MUNZNER T.: Steerable, Progressive Multidimensional Scaling. In *IEEE Symposium on Information Visualization* (Oct. 2004), pp. 57–64. ISSN: 1522-404X. doi:10.1109/INFVIS.2004.60. 2

[XESV97] XIA J., EL-SANA J., VARSHNEY A.: Adaptive real-time level-of-detail based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 171–183. doi:10.1109/2945.597799. 1

[ZGC*17] ZGRAGGEN E., GALAKATOS A., CROTTY A., FEKETE J.-D., KRASKA T.: How Progressive Visualizations Affect Exploratory Analysis. *IEEE Transactions on Visualization and Computer Graphics 23*, 8 (Aug. 2017), 1977–1987. doi:10.1109/TVCG.2016.2607714. 2

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum 34*, 2 (May 2015), 667–681. doi:10.5555/2816723.2816781. 5