ScaleFree: Dynamic KDE for Multiscale Point Cloud Exploration in VR

Lixiang Zhao[1] Fuqi Xie[1] Tobias Isenberg[2] Hai-Ning Liang[3] Lingyun Yu[1]*

[1] Xi'an Jiaotong-Liverpool University, Suzhou, China [2] Université Paris-Saclay, CNRS, Inria, LISN, France
[3] Hong Kong University of Science and Technology (Guangzhou), China

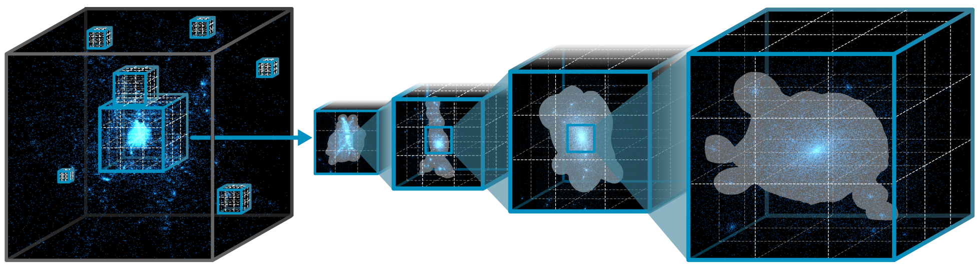

Fig. 1: ScaleFree dynamically recomputes density fields on the GPU, enabling on-the-fly density estimation of regions of interest (ROI) at any scale and position within massive multiscale point clouds in VR.

Abstract

We present ScaleFree, a GPU-accelerated adaptive Kernel Density Estimation (KDE) algorithm for scalable, interactive multiscale point cloud exploration. With this technique, we cater to the massive datasets and complex multiscale structures in advanced scientific computing, such as cosmological simulations with billions of particles. Effective exploration of such data requires a full 3D understanding of spatial structures, a capability for which immersive environments such as VR are particularly well suited. However, simultaneously supporting global multiscale context and fine-grained local detail remains a significant challenge. A key difficulty lies in dynamically generating continuous density fields from point clouds to facilitate the seamless scale transitions: while KDE is widely used, precomputed fields restrict the accuracy of interaction and omit fine-scale structures, while dynamic computation is often too costly for real-time VR interaction. We address this challenge by leveraging GPU acceleration with k-d-tree-based spatial queries and parallel reduction within a thread group for on-the-fly density estimation. With this approach, we can recalculate scalar fields dynamically as users shift their focus across scales. We demonstrate the benefits of adaptive density estimation through two data exploration tasks: adaptive selection and progressive navigation. Through performance experiments, we demonstrate that ScaleFree with GPU-parallel implementation achieves orders-of-magnitude speedups over sequential and multi-core CPU baselines. In a controlled experiment, we further confirm that our adaptive selection technique improves accuracy and efficiency in multiscale selection tasks.

Index Terms: Multiscale visualization, kernel density estimation, interactive data selection, interactive data navigation, VR/AR/MR.

1 Introduction

Advances in scientific computing have produced datasets that are not only larger in size but also richer in multiscale structural complexity. For example, modern cosmological simulations produce billions of particles, with details that span several orders of magnitude, to capture the formation and evolution of cosmic structures [54, 55]. The fact that meaningful features often emerge at different scales requires analysts to fluidly transition between global and local perspectives—for instance, from cosmic filaments and clusters to fine-grained substructures in astronomical point clouds [28, 55]. To understand the dynamics of structure formation, scientists need to visualize and explore the three-dimensional spatial patterns across multiple scales. Immersive environments delivered through virtual, augmented, and mixed reality head-mounted displays (HMDs) have emerged as superior tools for interpreting complex spatial relationships. Prior research has shown that immersive visualizations can enhance users' understanding of complex spatial structures, including volume data [38], astronomical data [74], and node-link diagrams [27, 60]. These benefits are particularly evident in navigation [53] and 3D manipulation tasks [8], where frequent viewpoint changes are required to inspect spatial structures. The immersive environment not only provide stereoscopic perception of 3D structures but also support seamless transitions between global overviews and localized detail exploration. For instance, a user may zoom in to inspect a galaxy cluster and then zoom out to contextualize its dynamics within the surrounding large-scale structures. This process requires not only an immersive environment, such as VR, but also high-performance scientific computing and visualization systems capable of supporting free navigation across space, time, and scale.

Achieving smooth transitions and effective exploration across multiple scales remains challenging, however, in particular when data must be transformed into an alternative representation to support further analysis. For example, KDE [19] has been widely used in point cloud visualization to uncover clusters and structural patterns in large multiscale datasets by converting discrete points into a continuous scalar density field. Such density fields are fundamental to spatial interaction, as they provide a robust basis for feature detection [12,15,45,50] and for constructing selection volumes [68,69,75], thereby enabling users to identify, isolate, and manipulate regions of interest. Nonetheless, computing scalar density fields on the fly is computationally expensive and often fails to guarantee real-time interaction. Prior density-based selection techniques for point cloud exploration [68, 69, 75] have typically focused on selection interaction itself and relied on precomputed, single-scale density fields, rather than targeting dynamic multiscale recomputation during interaction. Such precomputation, however, implies that the density fields are static during exploration, constraining selection accuracy to the resolution defined *a priori* and leaving finer-scale structures unrepresented. This limitation becomes particularly pronounced in VR. Unlike desktop systems, VR presents users with a single, continuous egocentric view of the data space, where navigation,

*Corresponding author: Lingyun Yu (lingyun.yu@xjtlu.edu.cn).

selection, and scale transitions are tightly coupled with head and body movement. Users frequently shift their focus between global overviews and localized regions, creating a strong demand for density fields that adapt immediately to scale and perspective changes. At the same time, VR imposes strict real-time constraints due to high-frequency stereoscopic rendering and low-latency interaction requirements. These challenges motivate us to explore density estimation techniques that can be dynamically computed at varying scales during real-time immersive exploration.

We thus introduce ScaleFree, a fast, adaptive, and scalable KDE algorithm specifically optimized for interactive point cloud data analysis. Our method leverages GPU acceleration to perform on-the-fly density estimation, enabling scalar fields to be recomputed dynamically as users shift their focus across scales. By maintaining density fields that adapt repeatedly rather than relying on precomputed representations, ScaleFree facilitates more precise and responsive support for core exploration tasks such as *spatial selection*, *feature localization*, and *multiscale navigation* in large point cloud datasets. To evaluate the role of adaptive KDE in interactive VR exploration, we adopt a multi-layered evaluation strategy. We first compare our GPU-based implementation against CPU baselines to establish real-time feasibility, and then evaluate how integrating adaptive KDE affects selection effectiveness through technique demonstrations and a controlled user study, measuring accuracy, efficiency, and workload against state-of-the-art VR selection techniques. Specifically, we conducted a user study with 24 participants comparing three *selection techniques* that employ different density estimation approaches, including precomputed single-resolution density fields (PS), precomputed multi-resolution mipmap density fields (PM), and our dynamic ScaleFree approach. Results show that selections made with ScaleFree were faster and more accurate than those using precomputed density fields. Participants also reported lower workload and expressed a stronger preference for ScaleFree when conducting multiscale exploration tasks in VR. In summary, our GPU-accelerated adaptive KDE makes the following main contributions to large-scale spatial data exploration in immersive environments:

- **Adaptive and smooth scale transitions:** enabling continuous transitions across scales without disruptive visual artefacts or noticeable delay (demonstrated in both selection Sec. 4.1 and navigation Sec. 4.2 scenarios).
- **Stable viewpoint-driven exploration:** supporting smooth, orientation-preserving transitions between overview and localized perspectives across scales during exploration of complex datasets (demonstrated in the navigation scenario, Sec. 4.2).
- **Fast and responsive computation:** improving selection accuracy and efficiency under varying scales, thereby supporting precise spatial interaction in immersive analytics (shown in the user study results, Sec. 5).

2 Related work

Our work builds on prior research in multiscale immersive exploration and density estimation methods that support such exploration.

2.1 Multiscale Exploration in Immersive Environment

Multiscale exploration requires users to operate across different levels of scale, ideally without relying on explicit mode switches or commands. Prior studies have demonstrated the advantages of immersive environments in diverse scientific domains such as astronomy [25, 52], geography [17], biology [37], and architecture [1, 40]. A widely adopted technique for multiscale navigation in immersive environments relies on a *World-in-Miniature* (WIM) [56], which presents both the life-size virtual environment and a hand-held, scaled-down replica that functions as an additional viewport. This dual representation enables users to interact with the environment at multiple scales: they can directly manipulate objects via the miniature, specify regions of interest for navigation, and then seamlessly "jump" to the corresponding areas in the full-scale world. Subsequent variations of WIM, such as Scalable WIM [46] and Scaled and Scrolling WIM [63], extended the metaphor by introducing panning and rescaling of the miniature, facilitating navigation across broader spatial ranges and more precise exploration of fine-grained structures, overcoming the scale and resolution constraints of the original WIM. Similarly, *target-based* and *steering-based* approaches, such as the magnifying glass metaphor by Kopper et al. [36], enable navigation through predefined discrete scale levels. Although these methods provide overview-detail support, however, they often fail to preserve users' sense of scale when "jumping" across scale levels.

To address this issue, Zhao et al. [74] linked multiple WIMs into a *hierarchical structure*, allowing users to seamlessly "jump" between different scales. Similarly, Bacim et al. [2] incorporated a hierarchical map to aid travel and wayfinding in anatomy applications. Moving beyond hierarchical structures, *label-based* approaches employ meaningful anchors that reflect data organization, as in Kouřil et al.'s [37] approach, who used active labels to browse hierarchical molecular visualizations. Nevertheless, these methods all depend on well-defined targets and hierarchical algorithms to compute and represent scale levels, with transitions typically occurring in either an end-to-end or step-by-step manner. Similar transition approaches are found in molecular visualization [29, 30], where hierarchical molecular structures provide clearly defined scale levels. By contrast, we focus on unstructured and massive scientific datasets, such as astronomical point clouds, where predefined or meaningful hierarchies are often absent. In such scenarios, "jumping" approaches are less straightforward—we need more flexible techniques that support navigation and interaction at arbitrary scales.

2.2 Key Aspects in Multiscale Immersive Exploration

Building on prior work in multiscale immersive exploration, we next synthesize key design aspects and limitations identified across existing systems. In particular, we focus on challenges related to fast and scalable computation, which motivate the need for adaptive density estimation techniques reviewed in the following subsection.

Scalable and flexible exploration. Effective multiscale exploration should enable users to select or navigate any arbitrary region within an object or space, as features may appear at any scale. When hierarchical information is not explicitly available, such as in astronomical point cloud data [28, 55] or additive manufacturing objects [16, 35] with dense, homogeneous internal structures, localization and navigation become particularly challenging. In such cases, continuous zooming enables more flexible and fluid multiscale exploration. To address this issue, Pavanatto et al. [43] proposed a progressive refinement approach, facilitating, step-by-step, focused inspections via a selection box. Our work shares this motivation but takes a different path: rather than refining predefined regions, we enable users to focus on relatively dense areas across diverse scales. This approach, in turn, requires a dynamic computation of density fields to adaptively support the exploration.

Adaptive and smooth transition. In free exploration, adaptive transitions are essential to reduce disorientation and preserve a natural sense of immersion in multiscale environments. To ensure continuity, techniques should dynamically adjust factors such as computational scale, navigation speed [1, 58], and the visualization's and users' relative scale [1, 59]. Traversing multiscale data, for instance, often requires adaptive "flying" speeds to balance comfort and continuity. Ware and Fleet [58] addressed this point by introducing continuous depth sampling to modulate the navigation speed, while Argelaguet et al. [1] extended the concept with an adaptive navigation technique that facilitates direct camera control, while it also automatically adjusts speed and environmental scale based on object distance and optical flow. More recently, Weissker et al. [59] proposed *teleportation-based* multiscale travel methods that integrate scale adjustments directly into the teleportation process. All this work highlights the need for approaches that enable users to transition smoothly through unstructured multiscale data, with navigation, visualization, and interaction parameters continuously adapted to the new scale, all of which we also strive for in our work.

Fast and responsive computation. Finally, achieving these interaction designs in practice requires scalable computational support: Techniques must remain efficient and responsive when applied to large, complex datasets, ensuring real-time performance. One common approach is to introduce hierarchies of reduced objects or resolutions to balance rendering and interaction costs. Gansner et al. [26], for example, precomputed a hierarchy of coarsened graphs that could be combined on-the-fly for visualization. For massive unstructured datasets such as point clouds, density estimation methods like KDE form the foundation for feature detection, region-of-interest selection, and navigation across scales. The key challenge, however, is ensuring that KDE can be computed on-the-fly to support such flexible and adaptive interactions across scales—on which we focus in our work and for which we review past work next.

2.3 Kernel density estimation (KDE)

KDE [9, 19] is a widely used method for computing data point distributions, which has been applied in diverse domains such as materials science [48], astronomy [70, 71], geoscience [44], ecology [24], and traffic accident studies [65]. Its core idea is that each data point is represented by a probability distribution (kernel; usually Gaussian [49] or Epanechnikov [22] ones) centered at its position, with the bandwidth controlling its spatial influence. Summing all kernels yields a continuous density field.

A key computational challenge in KDE lies in selecting the bandwidth, which controls the level of smoothing. The bandwidth can be specified as a constant (fixed KDE) or adapted to local point densities (adaptive KDE). Fixed KDE [7, 22] applies uniform smoothing across the dataset. While it is computationally less expensive, it often oversmooths dense regions and undersmooths sparse ones, limiting its accuracy in heterogeneous datasets. In contrast, adaptive KDE mitigates this issue by varying the bandwidth according to local density [7, 51]: points in sparse areas are assigned larger bandwidths, while those in dense regions receive smaller ones. This adaptive strategy yields more meaningful estimates in many applications, but comes at a high computational cost. For example, Brunsdon [7] proposed a cross-validation-based method for determining spatially adaptive bandwidths, but its two-dimensional parameter search makes it inefficient for large datasets. For high-dimensional data, researchers proposed hashing-based estimators [3] to achieve sublinear query time and linear space and preprocessing time.

KDE has been widely used in data visualization, with prior work exploring the algorithmic and system-level optimizations. Examples include GPU-accelerated KDE [39] for interactive streaming visualization, efficient kernel density techniques [13] for large-scale 2D hotspot visualization, adaptive GPU-based KDE methods [73] that combine algorithmic optimizations with parallel computation, and more recently, prefix-based spatiotemporal KDE frameworks [14] for large-scale, high-resolution density visualization. While these advanced KDE techniques offer improved theoretical efficiency and scalability and are tailored to different problem settings, our work focuses on immersive interaction integration and GPU-parallel implementation to support real-time, interactive multiscale exploration.

KDE has also been leveraged to support in the context of immersive analytics. Prouzeau et al. [47] introduced a VR-based technique that uses haptic feedback to convey KDE-derived density information, helping users identify occluded features in dense point cloud data. Zhao et al. [75] employed KDE in immersive spatial selection techniques, enabling users to define selection regions based on density variations rather than individual points. These approaches, however, rely on precomputed single-scale density fields, limiting their ability to capture fine substructures across multiple levels of detail. These efficiency limitations have motivated the exploration of high-performance computing strategies. An effective interaction with large-scale datasets, however, especially in immersive environments, requires not only scalability but also real-time, on-the-fly computation, which was not explicitly addressed in prior research. To this end, we formulate a GPU-parallel algorithm based on a modified Breiman kernel density estimation method with a finite-support, adaptive Epanechnikov kernel, and further propose a parallel optimization method to accelerate adaptive bandwidth calculation, together improving the efficiency of the entire KDE pipeline. To validate our approach, we demonstrate its use in two scientific data exploration scenarios and evaluate its effectiveness through both performance experiments and empirical user studies.

3 ScaleFree: A dynamic KDE technique

We begin our discussion with an overview of the KDE algorithm used in this work (Sec. 3.1), then present ScaleFree, our GPU-accelerated KDE for multiscale analysis (Sec. 3.2), and finally report on experiments that evaluate its performance (Sec. 3.3).

3.1 Kernel density estimation

KDE methods produce a smooth, continuous density field by distributing the contribution of each particle over a larger volume using a smoothing kernel function. This kernel function describes the spatial influence of each particle, assigning higher weights to locations closer to the particle center and lower weights to more distant locations. To estimate the scalar density field, similar to past works [68, 69, 75], we apply the modified Breiman kernel density estimation method (MBE) with a finite-support adaptive Epanechnikov kernel [23, 61] as follows: The whole dataset or region of interest is enclosed within a bounding box B and discretized into a uniform grid of resolution **res** (e. g., 128^3), with grid nodes located at positions $\mathbf{r}^{(n)}$, where n denotes the node index. For each spatial axis $k \in \{x, y, z\}$, we calculate the smoothing length ℓ as

$$\ell_k = \frac{2\left(P_k^{(80)} - P_k^{(20)}\right)}{\log N}. \tag{1}$$

Here, N is the number of particles in the box B, and $P_k^{(q)}$ represents the q^{th} percentile of the coordinates along axis k. For the grid node at position $\mathbf{r}^{(n)}$, we calculate the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$ as:

$$\rho_{\text{pilot}}(\mathbf{r}^{(n)}) = \frac{15}{8\pi N} \frac{1}{\ell_x \ell_y \ell_z} \sum_j E(\|\mathbf{r}^{(n;j)}\|), \tag{2}$$

with

$$\mathbf{r}_k^{(n;j)} = (\mathbf{r}_k^{(n)} - \mathbf{r}_k^{(j)})/\ell_k, \tag{3}$$

where $\mathbf{r}^{(j)}$ is the the j^{th} particle's position and $E(x)$ is the Epanechnikov kernel:

$$E(x) = \begin{cases} 1 - x^2, & |x| < 1, \\ 0, & |x| \geq 1. \end{cases} \tag{4}$$

The function $E(x)$ implies that a particle contributes to pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$ only if the particle $\mathbf{r}^{(j)}$ lies within the ellipsoid centered at the node position $\mathbf{r}^{(n)}$, with semi-axes ℓ_x, ℓ_y, and ℓ_z. This ellipsoidal restriction effectively limits the kernel bandwidth, ensuring that the resulting density estimate captures the local particle distribution. Then, we compute the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(j)})$ at the position of the j^{th} particle using multi-linear interpolation with respect to the densities on nearby nodes. Next, we compute the mean pilot density *aveDen* of all particles:

$$aveDen = \sum_j \rho_{\text{pilot}}(\mathbf{r}^{(j)}) \tag{5}$$

Algorithm 1: Fast and Scalable GPU-accelerated KDE

```
function FSKDE(kdTree, particles, nodes, res, l, s, density):
    Input: kdTree—k-d tree spatial indexing structure
    Input: particles—array of positions of all particles
    Input: nodes—array of positions of all grid-nodes
    Input: res—resolution of the grid (e.g. 64 × 64 × 64 )
    Input: l—array of smoothing length for every particle
    Input: s—node spacing along x, y, z axis
    Output: density—array of pilot density at grid-nodes
    Dispatch ( PDE (kdTree, particles, nodes, res, l, density), res_x/PDE_tx, res_y/PDE_ty, res_z/PDE_tz );
    Dispatch ( ASL (particles, l, s, density), particles.count/ASL_tx, 1, 1);
    Dispatch ( FDE (particles, nodes, res, l, density), res_x/FDE_tx, res_y/FDE_ty, res_z/FSDE_tz );
```

We then update the j-th particle's smoothing length by

$$\ell_k^{(j)} = \min\left(\ell_k \cdot \left(\frac{aveDen}{\rho_{\text{pilot}}(\mathbf{r}^j)}\right)^{\frac{1}{3}}, 5 \cdot s_k\right), \tag{6}$$

where ℓ_k is the initial smoothing length along axis k, s_k denotes the distance between adjacent nodes along the k-axis. To prevent the kernel from becoming excessively large—which could increase computational cost and overly smooth the density—we apply an upper bound of $5 \cdot s_k$. Then, we re-evaluate the density on all the nodes based on the updated smoothing length. For the grid node at position $\mathbf{r}^{(n)}$, we calculate the final density $\rho(\mathbf{r}^{(n)})$ as:

$$\rho(\mathbf{r}^{(n)}) = \frac{15}{8\pi N} \sum_j \frac{1}{\ell_x^{(j)} \ell_y^{(j)} \ell_z^{(j)}} E(\|\mathbf{r}^{(n;j)}\|), \tag{7}$$

with

$$\mathbf{r}_k^{(n;j)} = (\mathbf{r}_k^{(n)} - \mathbf{r}_k^{(j)}) / \ell_k^{(j)}, \tag{8}$$

Finally, we calculate the density $\rho(\mathbf{r})$ at any position $\mathbf{r}$ via trilinear interpolation of the surrounding grid nodes.

3.2 ScaleFree: Fast and scalable KDE on GPU

To enable on-the-fly density evaluation, we implement the KDE pipeline on the GPU with key optimizations (Fig. 11 in Suppl. A). Our GPU parallel implementation follows HLSL conventions, with background on the GPU programming model provided in Suppl. D. Prior to GPU execution, we perform essential preprocessing on the CPU: we determine the smoothing length (Eq. 1) and create a k-d tree spatial index for the point cloud dataset to accelerate future neighborhood queries. We then upload the input data (k-d tree, smoothing length, particle positions, grid node positions, grid resolution, and node spacing) to GPU global memory. In the GPU computation pipeline, as illustrated in HLSL-style pseudo-code in Alg. 1, we dispatch three compute kernels, each handling a distinct density estimation stage:

- In **pilot density estimation** (Alg. 2, Sec. 3.2.1) we compute a pilot density field at grid nodes by a uniform smoothing length.
- In **adaptive smoothing length** (Alg. 3, Sec. 3.2.2) we update per-particle smoothing lengths based on average pilot density.
- In **final density estimation** (Alg. 4, Sec. 3.2.3) we recompute the density field using the adaptive smoothing lengths.

3.2.1 Pilot density estimation (PDE)

To start, we compute the pilot density at each grid node by accumulating the contributions from neighboring particles. We adopt the *gather* approach [32], where each thread processes a grid node and collects contributions from neighboring particles, rather than a *scatter* approach, where each thread processes a particle and distributes its contribution to surrounding nodes. This way we avoid costly atomic operations and improve the performance and scalability for our large-scale particle datasets.

Algorithm 2: Pilot Density Estimation

```
numthreads(PDE_tx, PDE_ty, PDE_tz)
function PDE(kdTree, particles, nodes, res, l, density):
    Input: kdTree, particles, nodes, res, l
    Output: density
    uint3 DTid : SV_DispatchThreadID;
    uint idx = DTid_x + DTid_y · res_x + DTid_z · res_x · res_y;
    if idx ≥ nodes.count then
        return;
    float3 query = nodes[idx];
    float range = max(l[0]_x, l[0]_y, l[0]_z);
    uint retNbrs; // the number of returned neighbor particles
    float retDists[]; // the array of neighbor–query distances
    KDTreeSearch(kdTree, query, range, retNbrs, retDists);
    float sum = 0;
    for i in 0 : retNbrs−1 do
        float3 r = float3(retDists[i]_x / l[0]_x, retDists[i]_y / l[0]_y, retDists[i]_z / l[0]_z);
        sum += max(0, 1 − ||r||^2)
        i++;
    density[idx] = 15·sum / (8π·particles.count·l[0]_x·l[0]_y·l[0]_z);
```

We dispatch the **PDE** kernel (Alg. 2) with $\frac{\mathbf{res}_x}{PDE_{tx}} \times \frac{\mathbf{res}_y}{PDE_{ty}} \times \frac{\mathbf{res}_z}{PDE_{tz}}$ thread groups, each consisting of $PDE_{tx} \times PDE_{ty} \times PDE_{tz}$ threads, to cover the entire grid domain. For each thread with global index idx, we evaluate the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(idx)})$ at node position $\mathbf{r}^{(idx)}$ using Eq. 2. Since only particles within the kernel support contribute to the density (Eq. 4), we first identify candidate particles through a spherical range query on the k-d tree, centered at $\mathbf{r}^{(idx)}$ with radius $\max(\ell_x, \ell_y, \ell_z)$, i. e., the largest of the three kernel semi-axes ℓ_x, ℓ_y, and ℓ_z. This query allows us to discard of particles outside the Epanechnikov kernel. In this way, for M nodes and N particles, we reduce the complexity of the original KDE algorithm from $O(MN)$ to $O(M\sqrt{N})$ (where k-d tree searching has complexity $O(\sqrt{N}+K)$, with K the number of neighbors returned [34]). Finally, we compute the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(idx)})$ at node position $\mathbf{r}^{(idx)}$ using Eq. 2 and store it in the idx-th position of density buffer.

3.2.2 Adaptive smoothing length (ASL)

Next, we update the smoothing length of each particle based on the result of pilot density estimation. The goal is to assign larger smoothing lengths in sparse regions and smaller ones in dense regions, thereby yielding a smoothly density field in the subsequent stage. Since the update is particle-based, we assign one thread per particle to compute its adaptive smoothing length, which provides a natural and efficient mapping for parallel execution.

We dispatch the **ASL** kernel (Alg. 3) with $\frac{N}{ASL_{tx}} \times 1 \times 1$ thread groups, each containing $ASL_{tx} \times 1 \times 1$ threads, to cover the entire grid domain. For the thread with global index idx, we compute the adaptive smoothing length of the idx^{th} particle as follows.

First, we compute the pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(idx)})$ of the idx^{th} particle using trilinear interpolation and stores the result as `pDen = TrilinearInterp(particles[idx], density)`.

Second, we calculate the arithmetic mean *aveDen* of the pilot densities across every particle in box B, using Eq. 5 for efficient aggregation. To accelerate this step, we design a hierarchical computational method (HCM) that employs parallel reduction with shared memory, thereby reducing costly global memory accesses. Specifically, within each thread group, we first allocate a shared memory array `sharedDen`, where each thread stores its locally computed pilot density. We then perform a parallel reduction within the group to compute its average density, which is written to a global memory array `groupAveDen` with N/ASL_{tx} entries. Then, we obtain the overall mean pilot density by averaging all values in `groupAveDen`. Specifically, for thread with global index idx, along with other threads in the same thread group (group index GID), we proceed as follows:

Algorithm 3: Adaptive Smooth Length

$numthreads(ASL_{tx}, 1, 1)$
function ASL($particles, l, s, density$):
  Input: $particles, l, s, density$
  Output: l
  $uint\ GI$: SV_GroupIndex;
  $uint3\ GID$: SV_GroupID;
  $uint3\ DTid$: SV_DispatchThreadID;
  $uint\ idx = DTid_x$;
  $float\ groupAveDen[\frac{particles.count}{ASL_{tx}}]$;
  $groupshared\ float\ sharedDen[ASL_{tx}]$;
  if $idx \geq particles.count$ **then**
    $return$;
  $float\ pDen = TrilinearInterp(particles[idx], density)$;
  $sharedDen[GI] = pDen$;
  $GroupMemoryBarrierWithGroupSync()$;
  // parallel reduction
  for $i\ in\ ASL_{tx}/2 : 0$ **do**
    if $GI < i$ **then**
      $sharedDen[GI] += sharedDen[GI+i]$;
      $GroupMemoryBarrier()$;
    $i >>= 1$;
  if $GI == 0$ **then**
    $groupAveDen[GID_x] = sharedDen[0]/ASL_{tx}$;
  $WaitForAllThreads()$;
  $float\ aveDen = groupAveDen.Average()$;
  $l[idx]_x = \min\left(l[idx]_x \cdot \left|\frac{aveDen}{pDen}\right|^{\frac{1}{3}}, 5 \cdot s_x\right)$;
  $l[idx]_y = \min\left(l[idx]_y \cdot \left|\frac{aveDen}{pDen}\right|^{\frac{1}{3}}, 5 \cdot s_y\right)$;
  $l[idx]_z = \min\left(l[idx]_z \cdot \left|\frac{aveDen}{pDen}\right|^{\frac{1}{3}}, 5 \cdot s_z\right)$;

1. Thread idx stores its pilot density $pDen$ into the shared memory array `sharedDen` at its local index GI (idx^{th} thread's index within it's thread group), i. e., `sharedDen[GI] = pDen`.
2. All threads in group GID wait until every thread in the same group has updated `sharedDen` with its calculated pilot density, i. e., `GroupMemoryBarrierWithGroupSync()`.
3. All threads in group GID perform a parallel reduction on `sharedDen` to accumulate all density values. After execution, the total value is located at 0^{th} position in `sharedDen` array.
4. The 0^{th} thread in group GID computes the average value of `sharedDen` and then writes it to the global memory array `groupAveDen` at index GID, i. e., `groupAveDen[GID_x] = sharedDen[0]/ASL_tx`.
5. All threads wait until every group's 0^{th} thread has written its result to `groupAveDen`, i. e., `WaitForAllThreads()`.
6. Thread idx computes the global average pilot density $aveDen$ as the mean of all elements in `groupAveDen`, i. e., `aveDen = groupAveDen.Average()`.

Third, thread idx computes the smoothing length of the idx^{th} particle using Eq. 6 and stores it in the smoothing length buffer.

3.2.3 Final density estimation (FDE)

Finally, we recompute the density at each node using the adaptive smoothing length of its contributing particles. We dispatch the **FDE** kernel (Alg. 4), where each thread computes the final density at a single grid node. To cover the grid, we launch $\frac{\mathbf{res}_x}{FDE_{tx}} \times \frac{\mathbf{res}_y}{FDE_{ty}} \times \frac{\mathbf{res}_z}{FDE_{tz}}$ thread groups, each containing $FDE_{tx} \times FDE_{ty} \times FDE_{tz}$ threads. The thread with global index of idx evaluates the final density $\rho(\mathbf{r}^{(idx)})$ at node $\mathbf{r}^{(idx)}$ using Eq. 7 and stores the result to the density buffer at position idx. After all threads have completed, we transfer the density buffer from GPU to CPU memory.

Algorithm 4: Final Density Estimation

$numthreads(FDE_{tx}, FDE_{ty}, FDE_{tz})$
function FDE($particles, nodes, res, l, density$):
  Input: $particles, nodes, res, l$
  Output: $density$
  $uint3\ DTid$: SV_DispatchThreadID;
  $uint\ idx = DTid_x + DTid_y \cdot res_x + DTid_z \cdot res_x \cdot res_y$;
  if $idx \geq nodes.count$ **then**
    $return$;
  $float\ sum = 0$;
  for $i\ in\ 0 : particles.count - 1$ **do**
    $float3\ r$;
    $r_x = \frac{nodes[idx]_x - particles[i]_x}{l[i]_x}$;
    $r_y = \frac{nodes[idx]_y - particles[i]_y}{l[i]_y}$;
    $r_z = \frac{nodes[idx]_z - particles[i]_z}{l[i]_z}$;
    $sum += \frac{\max(0, 1 - \|r\|^2)}{l[i]_x \cdot l[i]_y \cdot l[i]_z}$;
    $i++$;
  $density[idx] = \frac{15 \cdot sum}{8\pi \cdot particles.count}$;

3.3 Performance analysis

Before we report our empirical results, we briefly analyze the algorithmic complexity of ScaleFree to highlight its fundamental sources of speedup. These optimizations target the density field computation, which requires recomputation whenever the view changes (e. g., transitioning to a new scale). For pilot density computation, we use a k-d tree to reduce neighborhood search complexity from $O(MN)$ for M grid nodes and N particles to $O(M\sqrt{N})$. To achieve an adaptive smoothing length, we employ a hierarchical method that utilizes thread-group shared memory and parallel reduction, thereby lowering the aggregation complexity from $O(N)$ to $O(\log N)$ within each group of size N. Combined with GPU parallelism, these optimizations enable ScaleFree to scale efficiently to hundreds of thousands of particles and deliver low-latency density field recomputation. To demonstrate this performance in practice, next we evaluate our approach on astronomical point cloud data.

Design. To assess the *efficiency* of ScaleFree, following prior work [73], we implemented three density estimation strategies: a sequential version on a single CPU core (SC) as baseline, a parallel version on a multi-core CPU (MC), and the GPU-parallel version (ScaleFree) as described in Sec. 3.2. These CPU-based baselines provide a controlled reference that allows us to establish real-time feasibility and to attribute performance differences specifically to GPU parallelization and dynamic recomputation, rather than to algorithmic variation. We compared execution time across the three strategies using the acceleration factor (AF) [72] as a performance metric, defined as $AF = T_{baseline}/T_{target}$. We report the average execution time over 10 runs for all three strategies, with GPU-parallel ScaleFree results excluding CPU–GPU data transfer.

Environment and implementation. We conducted the experiment on a workstation running Windows 11 on Intel's 13^{th} generation Core™ i9-13900KF processor (3.0 GHz, 64 GB RAM) and an NVIDIA GeForce RTX 4090 GPU (24 GB of memory) for acceleration. We implemented ScaleFree in Unity3D, with GPU computations realized through compute shaders. We set the resolution res of box B to 64^3 (262k nodes) and the configuration of the thread group dimensions (i. e., $PDE_{tx}, PDE_{ty}, PDE_{tz}, FDE_{tx}, FDE_{ty}, FDE_{tz}$) as 8 and ASL_{tx} as 1024. In addition, we implemented the sequential CPU strategy (SC) using C# and the multi-core CPU strategy (MC) using C#'s Task Parallel Library (TPL).

Datasets. We used three cosmological point cloud datasets: Nbody 1 (76k points, Fig. 2(a)), Nbody 2 (164k points, Fig. 2(b)), and Filament (442k points, Fig. 2(c)). The two N-body datasets feature a dense central cluster surrounded by smaller ones [54], while the filament dataset depicts a cosmic web with thin filaments [55].

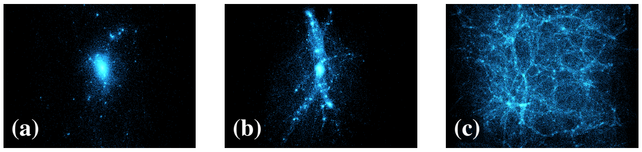

Fig. 2: Experiment datasets: (a) Nbody 1 (76k points), (b) Nbody 2 (164k points), and (c) Filament (442k points).

Table 1: Execution time (seconds) and acceleration factor of SC, MC (32 cores), and ScaleFree on a 64^3 grid (262k nodes).

metric	algorithm	Nbody1 (76k)		Nbody2 (164k)		filament (442k)	
		mean	std. dev.	mean	std. dev.	mean	std. dev.
execution time (s)	SC	7.689	0.161	58.373	6.643	395.675	12.458
	MC	1.542	0.041	7.244	0.264	48.514	1.567
	ScaleFree	0.042	0.002	0.119	0.003	0.309	0.011
acceleration factor	SC/ScaleFree	183.1	–	409.5	–	1280.5	–
	MC/ScaleFree	36.7	–	60.8	–	157.0	–

Efficiency. Our results (Tab. 1) show that ScaleFree ran substantially faster than SC and MC, with standard deviation consistently within 5% of the mean, indicating stable performance. As dataset size increases, ScaleFree becomes increasingly more efficient relative to CPU implementations, with the acceleration factor rising accordingly. For the 76K-point dataset discretized on a 64^3 grid, ScaleFree achieved an execution time of 0.042 s, corresponding to a sustained rate of about 20 FPS if executed continuously. Although the computation time does not meet the standard 60 FPS, density recomputation only needs to be triggered occasionally, when the view changes substantially. Moreover, the execution times of ScaleFree were generally in the order of 0.3–0.1s or substantially lower, so that the "illusion of animation" remains unaffected [11]. The overall runtime and system fluidity thus remain unaffected.

4 Multiscale Exploration with ScaleFree

We now discuss how our dynamic KDE approach contributes to improving large-scale scientific data exploration in VR.

4.1 Scalable selection technique

Selection is a fundamental operation for many subsequent data analyses [5, 62]. Maintaining both efficiency and accuracy, however, is particularly challenging in multiscale point cloud datasets due to their inherent features: highly complex structures, multiple levels of scale, occlusion, and unclear or overlapping boundaries. Designing selection techniques for multiscale point cloud data requires an understanding of user interaction. Users navigate across scales to find a suitable view that allows them to see targets clearly. For ease of use, however, they may also select point clouds even when not at the ideal scale. Selection must thus be supported anywhere and at any scale. As users move across scales, furthermore, they may not want to manually adjust selection parameters—they expect these adjustments to happen automatically. In sum, a selection technique should be dynamic, adaptive, accurate, and efficient, which is exactly what we can realize by fitting our dynamic KDE into the selection workflow. Workflow visualization available in supplementary materials.

At the start of an interactive session, the global view is initialized. As users navigate to a new scale, we recompute the density field on the GPU using our KDE (Sec. 3.2) and then transfer it back to the CPU for subsequent operations. Although we update the density field whenever the scale changes, the GPU pipeline enables rapid recomputation, ensuring smooth navigation.

When users initiate a selection, we invoke a GPU kernel to compute the selection volume. Specifically, once a selection is initialized, we determine a density threshold for the interacting area and transfer

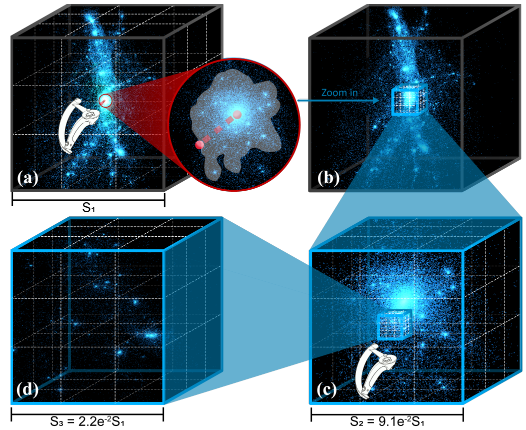

Fig. 3: Our progressive navigation technique.

it to the GPU. A kernel then executes the Marching Cubes algorithm [41, 64] to extract the selection volume as an iso-surface based on this threshold. We then select points within the extracted volume. Thanks to our optimized computation, the selection volume is generated rapidly. Even with the MeTAPoint technique [75], which detects density thresholds on-the-fly at the user's interaction point, users can immediately observe updates to the selection volume and assess whether the result aligns with their intent.

Overall, our framework supports adaptive and smooth scale transitions during selection by recomputing density fields on the fly to match users' interactive focus. This enables users to navigate and zoom across scales without noticeable delay. Since the density field is recomputed with our optimized KDE approach once after a view change and remains available for subsequent selections, the technique supports fluid interaction at any scale.

4.2 Progressive navigation technique

Beyond being a selection technique, in fact, ScaleFree also integrates navigation and zooming into a unified workflow. Navigation is a fundamental exploration task that relocates the viewpoint toward a region of interest while preserving spatial context. It has been widely studied in visualization research across different scenarios, such as gaining insights from volumetric [33] and abstract data [67]. Navigation techniques consist of several components, including wayfinding, travel, and context switching. In large multiscale point clouds, navigation becomes particularly challenging because salient structures are distributed across multiple scales, and dense spatial distribution introduces severe occlusion and clutter. These issues complicate wayfinding and travel, increase the number of travel steps that often occur across scales, and intensify context switching, making it easy for users to lose orientation and spatial context when moving across scales. Navigating such data in immersive environments is even more demanding, as the strong sense of presence can amplify both perceptual challenges and spatial disorientation. To address these challenges, we propose a scale-aware progressive navigation technique that enables users to explore multiscale point cloud data across scales. By defining a region of interest (ROI)—essentially a selection interaction that we have just described—our technique guides users through successive scales (i. e., a combined navigation and zooming interaction) to reach the most suitable viewpoint for observation. Next, we demonstrate how our dynamic KDE can fit into the navigation/zooming workflow to support this process. A workflow visualization is provided in Suppl. A.

An interactive session starts from a global view of the dataset. As before, we recompute the density field on the GPU using our KDE (Sec. 3.2) whenever the view changes substantially. To navigate/zoom to a specific area, users define an ROI with our scalable selection technique (Fig. 3(a), Sec. 4.1). Because we compute the density field at the current scale, the ROI is immediately identified and we can smoothly transition the viewpoint to the ROI's center (from Fig. 3(b) to (c)). We adjust the size of the new viewport to adequately cover the ROI (e. g., the identified selection volume), guiding users to an appropriate scale for observation. Together, this process supports stable viewpoint-driven exploration by preserving spatial orientation and contextual continuity across scale changes. While the current implementation focuses on the orientation consistency, a promising extension could also be to combine our scale-aware navigation with feature-aware techniques (e. g., [10,66]) to automatically compute an optimal view orientation, enabling users to travel across scales and arrive at the most informative perspective.

After each viewpoint update, we recompute the density field for the new scale, allowing users to effectively perform further ROI selections that guide subsequent navigation steps (as shown in Fig. 3(d)). This progressive "*targeting-travel-context switching*" workflow ensures smooth viewpoint transitions and multiscale navigation without the need for manual wayfinding. In addition, these travel steps can be recorded to enhance spatial awareness during progressive navigation. Overall, our dynamic KDE enables seamless, scale-aware navigation by continuously updating density fields to support progressive ROI selection and smooth view transitions.

5 User Study

To evaluate whether our scalable selection method meets the goals of being dynamic, adaptive, accurate, and efficient, we conducted a within-subjects controlled user study comparing it against two density-based strategies: one using a precomputed single resolution and the other using precomputed multiple resolutions. We focus on the selection interaction, as it serves as the foundation for both pure data selection (Sec. 4.1) and selection-based progressive data navigation/zooming (Sec. 4.2). We pre-registered our study (`osf.io/hfu6e`) and received IRB approval for the protocol (XJTLU University Research Ethics Review Panel, № ER-LRR-0010000120520250813002121).

5.1 Study Design

Participants. We recruited 24 unpaid participants (11 male, 13 female) from the local university, aged 19–30 years (M=23.25, SD=3.22). Among them, 21 were right-handed, two were left-handed, and one was ambidextrous. Twelve participants reported using VR at least once per week, 11 at least once per year, and one had no prior experience with VR devices. Furthermore, 16 participants had obtained a Bachelor's degree or higher. All participants had normal ($n = 16$) or corrected-to-normal ($n = 8$) vision and could clearly distinguish the colors we used in our study.

Apparatus. We used the Vive Pro 2 [6], a PC-based VR head-mounted display (HMD; 2448×2448 resolution per eye, 116° field of view, 120 Hz refresh rate). The study was carried out on a PC (Intel 13th Gen Core™ i9-13900KF processor, 3.0 GHz, 64 GB RAM and an NVIDIA GeForce RTX 4090 GPU, 24 GB of memory).

Datasets. We extracted five timesteps from a cosmological N-body simulation [54] and used them as our datasets. These datasets feature stellar clusters distributed across multiple scales, where zoomed-in views reveal progressively detailed substructures. Each dataset contained eight to ten target structures highlighted in yellow, distributed across different scales. Tasks began at varying scales, with all targets becoming visible as participants zoomed in. We show an example dataset with its target structures in Fig. 4, and other four datasets in the Suppl. C.

Task and Procedure. We instructed participants to select the yellow particles while avoiding the blue ones. Prior to the main experiment and after we had obtained their informed consent, we trained them on the MeTAPoint selection technique [75] using practice datasets. Unlike what Zhao et al. [75] did in their MeTAPoint study, we allowed participants to freely adjust scales (via zooming) and to perform selections at different levels. We also provided undo,

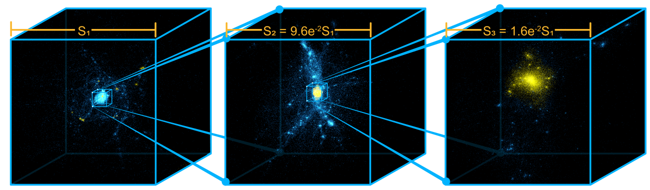

Fig. 4: One of the datasets we used in our study.

redo, and reset functions to restore the initial state if needed. In the main experiment, we asked participants to complete the selections as quickly and accurately as possible. We did not, however, provide suggestions on whether or when the selection was complete. After each condition, we asked participants to report their workload and fatigue using NASA's Task Load Index (TLX) [31]. At the end of all trials, we asked them to indicate their preferred technique and explain their choice, focusing on perceived fluency in multiscale selections. The interview questions are provided in Suppl. E. All participant responses are shared on OSF. A whole session lasted approx. 45 minutes.

Density Field Conditions. We asked our participants to perform all selections using the same density-based MeTAPoint method [75]: they initiated a selection by pointing at or near a target cluster and then dragged along its boundary while holding the VR controller trigger. The underlying density field computation, however, differed across approaches, resulting in three experimental conditions:

Precomputed Single-resolution (PS): The density field is precomputed once at a fixed resolution (64^3) before the task. During the task, participants interact with this static field, which remains unchanged across all scale levels.

Precomputed Multi-resolution (PM): The density field is precomputed at two resolutions (64^3 and 128^3). Users interact with the precomputed field interpolated by fields according to their scale level. We use two reference scales, S_{max} and S_{min}, which are predefined—at S_{min} the highest-resolution field is used, whereas at S_{max} the lowest-resolution field is employed. For the current scale S_c during exploration we determine the corresponding mipmap level $ml \in [0,1]$ by

$$ml = \frac{S_c - S_{min}}{S_{max} - S_{min}}. \tag{9}$$

and use ml to obtain the immediate density field via linear interpolation between the two fields with different resolutions.

Dynamic Resolution (DR): We compute the density field in real time via ScaleFree (Sec. 4.1). We set the field resolution to 64^3 and configure the thread group dimensions as PDE_{tx}, PDE_{ty}, PDE_{tz}, FDE_{tx}, FDE_{ty}, $FDE_{tz} = 8$, and $ASL_{tx} = 1024$.

While PS represents the conventional approach of using a fixed-resolution density field for density-based selection [68, 69, 75], PM further optimizes this strategy by adopting a mipmap-inspired hierarchy from computer graphics, enabling multiscale access through precomputed density levels without real-time recomputation. By comparing DR against these two representative paradigms, we assess the performance of dynamic KDE in selection tasks.

Design. We counterbalanced the three density field conditions using full permutation (in total six possible orders). We assigned the first six participants one order each based on P_{ID} mod 6, and repeated the scheme for every subsequent group of six. In total, 24 participants $\times$ 3 methods $\times$ 5 datasets yielded 360 trials.

Measures and Analysis. We recorded accuracy, completion time, as well as the transition times in the study. Given the criticism of NHST in the analysis of experimental data [4, 18, 20, 21], and APA's advice to seek other methods [57], we present our findings using estimation techniques that report effect sizes and confidence intervals instead of relying on p-value statistics.

Accuracy: Similar to Yu et al. [68, 69, 75], we evaluated the accuracy using two metrics: F1 and MCC (Matthews correlation

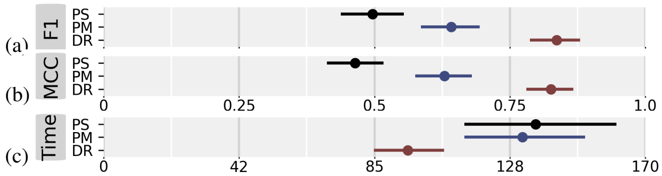

Fig. 5: The mean of (a) F1, (b) MCC, and (c) time, with 95% CIs.

coefficient). To calculate these, we identified true positives (TP; correctly selected particles), false positives (FP; incorrectly selected particles), false negatives (FN; target particles that were not selected), and true negatives (TN; correctly unselected particles). We defined precision as $P = TP/(TP+FP)$ and recall as $R = TP/(TP+FN)$, and calculated F1 as $\mathrm{F1} = 2\cdot(P\cdot R)/(P+R)$. While F1 reflects the harmonic mean of precision and recall, it does not account for TN. Thus, we also computed MCC, defined as:

$$\mathrm{MCC} = \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

We normalized all accuracy scores, and computed means and 95% bootstrap confidence intervals (CIs; $n = 24$).

Completion Time: We analyzed completion times using exact CIs on log-transformed data ($n = 24$). We report results as geometric means and demonstrate means comparisons as ratios.

5.2 Hypotheses

We formulated the following hypotheses based on the underlying principles of each density field condition:

H1: DR yields higher accuracy than PM and PS.
H2: PM yields higher accuracy than PS.
H3: DR requires less completion time than PM and PS.
H4: PM requires less completion time than PS.
H5: DR has a lower cognitive load than PM and PS.
H6: DR has a higher preference than PM and PS.

The rationale behind **H1** is that density fields in DR are dynamically adjusted to the current scale during navigation and zooming. At any scale, the density field is recomputed with a resolution 64^3, ensuring consistent accuracy. By contrast, PS and PM rely on precomputed density fields: when zoomed in, they use coarser grids that reduce precision. Nevertheless, PM integrates density fields at two scales rather than just one as in PS. Therefore, selections made with PM should be more accurate than those with PS, as stated in **H2**. The rationale behind **H3** is that DR provides the most accurate results, making selections easier for users to accept and thereby reducing completion time. Although the density fields in PS and PM are pre-computed and do not require on-the-fly computation, users may spend additional time refining their selections to achieve greater precision. A similar reasoning applies to **H4**: because PM yields more accurate results than PS, users are more likely to accept the selection outcomes with less refinement, resulting in shorter completion times. In addition, DR enables users to select target points accurately at any scale without the need to find an optimal view or repeatedly refine their selections, which reduces both effort and mental demand—thereby lowering cognitive load (**H5**). Consequently, users are more likely to prefer DR over PS and PM (**H6**).

5.3 Results

In Fig. 5 we show the mean completion times and two accuracy metrics (F1 and MCC) with 95% confidence intervals across density field conditions (Precomputed Single-resolution (PS), Precomputed Multi-resolution (PM) and Dynamic Resolution (DR)). We further provide the pairwise ratio with 95% confidence intervals in Fig. 6. The numerical values of average task completion times, two accuracy scores for each dataset and technique, and the numerical values of pairwise ratio are provided in Suppl. B. We now report statistical results in relation to our hypotheses.

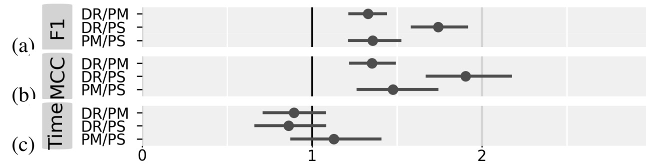

Fig. 6: Pairwise ratio of (a) F1, (b) MCC, and (c) time, 95% CIs.

Accuracy. For F1, a score of 1 indicates perfect performance and 0 the worst. For MCC, a score of 1 indicates perfect performance, while -1 represents the worst performance. As shown by the mean F1 scores (Fig. 5(a)) and MCC scores (Fig. 5(b)), DR outperformed both PM and PS in accuracy. The pairwise comparisons reveals that DR achieved 1.21–1.44× higher F1 score and 1.22–1.49× higher MCC than PM, and 1.58–1.92× higher F1 and 1.67–2.18× higher MCC than PS. **H1 is supported**. In addition, PM outperformed PS in both mean F1 and MCC. The pairwise comparisons show that PM achieved 1.21–1.53× higher F1 and 1.26–1.75× higher MCC than PS. Thus, **H2 is also supported**.

Completion Time. Fig. 5(c) shows that selections with DR required less completion time than PM and PS. The pairwise comparisons indicate that DR took 0.70–1.08× as long as PM and 0.66–1.08× as long as PS, suggesting that most of participants completed the tasks more quickly with DR. In a few cases, ratios were only marginally above 1.0, reflecting participants who finished marginally faster with PM or PS. We made related observations during the study: 4 participants ended the tasks early once they realized that they could not achieve satisfactory outcomes with PS or PM. These behaviors, however, do not affect the overall result. We confirm that **H3 is supported**. Our results also showed that PM had a slightly shorter mean completion time than PS. The pairwise comparisons indicate that PS took 0.87–1.40× longer than PM, suggesting that the advantage of PM over PS was modest rather than substantial. Thus, **H4 is not fully supported**.

Workload and Preference. As shown in Fig. 7, participants reported greater efficiency, lower frustration and time pressure, and reduced mental and physical effort with DR compared to PM and PS. Moreover, they ranked the conditions consistently ($DR > PM > PS$) across speed, accuracy, overall preference, and fluency (Fig. 8). Thus, **H5 and H6 are supported**.

Interview. In follow-up interviews, we asked participants whether they perceived delays during scale changes (where recomputation happens at the end of scale change) and selections across the three techniques. Most participants ($n = 20$) reported no noticeable recomputation delay with DR, while four noted slight delays only when the number of points in view was very large. These delays, however, did not affect their navigation experience. In contrast, a majority of participants ($n = 18$) identified PM as producing the most prominent lag: when using MeTAPoint to select large regions and generate extensive selection volumes, they observed noticeable drops frame rate. This performance degradation stems from the underlying characteristics of PM: higher-resolution grids increase the number of triangles needed to construct the selection volume and impose heavier computational demands when determining selection thresholds. These findings highlight scenarios where dynamic density fields offer substantial advantages over precomputed higher-resolution fields, particularly for sparse or large-scale datasets. In such cases, precomputed fields must rely on densely sampled grids to capture comparable detail, whereas dynamic estimation adapts density computation on demand, enabling the capture of fine-grained structures while maintaining smooth frame rates.

Summary. Improvements in F1 and MCC (**H1**) and reduced completion time (**H3**) show that our dynamic KDE improves both selection accuracy and efficiency over precomputed methods. By maintain-

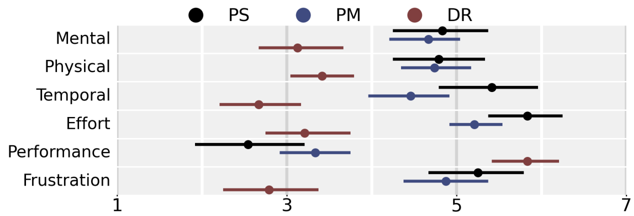

Fig. 7: User workload and performance. Error bars: 95% CIs.

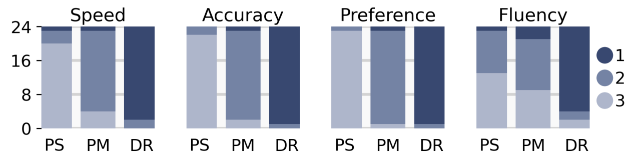

Fig. 8: Participant rankings on task performance and preference.

ing scale-accurate density fields, it enables high-fidelity selection with less effort and lower mental demand (**H5**), and was therefore preferred by participants (**H6**). These results highlight that on-demand recomputation is essential for precise immersive interaction, as precomputed methods often miss fine-scale details and require additional adjustments.

6 Discussion

In this section, we discuss how ScaleFree extends to support broader tasks, analyze the trade-offs between dynamic and precomputed density fields, and limitations and future directions.

6.1 Extending ScaleFree

Point cloud data often lacks a natural scale. Such datasets consist of unstructured points scattered unevenly across scales, sometimes with vast empty ranges, as in astronomical simulations. As a result, interaction and visualization approaches must help users orient themselves within scales and navigate across them. Such tasks require effective methods to compute key features, such as density, as the data changes—forming the starting point of our work. Multiscale data often contains fine details embedded within broader structures, creating large gaps between scales. A key feature of our dynamic KDE is that it allows users to explore subtle or complex structures on demand, with control remaining in the users' hands. For example, by dynamically adjusting resolution based on the interaction history and on the local data distribution, the navigation approach can guide users toward scale levels where meaningful features emerge. A future direction of research, however, is to determine how our KDE method can be adapted to better handle challenging cases such as if the data distribution dynamically changes across scales.

6.2 Trade-offs

We conducted performance experiments to evaluate ScaleFree, comparing its execution time against density estimation on single-core and multi-core CPUs. We also compared selection techniques using ScaleFree with those based on precomputed density fields in terms of accuracy and efficiency. These experiments highlight the trade-offs between dynamic and precomputed approaches, offering guidance for future researchers in selecting suitable solutions.

When designing an appropriate strategy, factors such as data scale, hardware capacity, and algorithmic complexity must be considered. Our results show that dynamic KDE consistently delivered higher accuracy and faster computation than precomputed density fields. We also found that multi-resolution density fields achieved better accuracy than single-resolution ones, though with only marginal gains in speed. This suggests that for datasets of moderate scale, precomputing a few density fields may be sufficient—especially when there is only limited memory available—since users can quickly access and switch values without recomputation. For datasets spanning multiple scales, however, dynamic KDE offers clear advantages by computing density fields on demand and supporting seamless multiscale exploration—yet these advantages come with a higher demand for GPU support and GPU/CPU memory and a more complex implementation. Still, an effective strategy is needed to determine when recomputation should occur, as users may notice delays during density field updates. To address this point, we trigger recomputation after users transition to a new scale. While users take time to observe the data at this scale, the system prepares the density field on the GPU, ensuring readiness for subsequent operations.

6.3 Limitations and Future Work

Several limitations remain that open avenues for future work.

First, although DR dynamically recomputes the density field at a fixed resolution of 64^3, fine details may still be missed. Although progressive navigation can reveal these features step by step at finer levels, it remains ineffective. Future work could adopt feature-aware resolution schemes that adapt to the underlying distribution, ensuring salient structures in the region of interest are faithfully represented.

Second, our dynamic KDE can be further optimized. Currently, recomputation is triggered even for minor scale changes, leading to unnecessary overhead during frequent zooming. To address this issue, we can introduce a threshold for scale changes, ensuring that recomputation occurs only when the change exceeds a meaningful level. This improvement would reduce redundant updates while maintaining responsiveness. Similarly, we could only compute the KDE for subsets of the dataset at any given zoom level (as opposed to the whole data space), such as the size of the viewport plus some margin around it. Then the recomputation for other regions of the data would be triggered when the user in their interactive navigation gets close to the boundary of the currently computed density field.

Third, the main time cost of the current recomputation delay in our selection technique lies in transferring the computed density field from the GPU back to the CPU. Future implementations could eliminate this bottleneck by migrating the selection algorithm, or any other interaction entirely onto the GPU, thereby avoiding density transfer from GPU to CPU.

Finally, this work is driven by the demands of immersive multiscale point cloud exploration and therefore focuses on GPU-parallel implementation and system-level optimization of adaptive KDE, rather than proposing new density estimation formulations. While recent KDE methods [14, 42] offer improved theoretical efficiency and memory usage, their integration into a real-time interactive VR pipeline remains an important direction for future work.

7 Conclusion

In this work, we present a GPU-accelerated adaptive KDE technique that dynamically recomputes density fields for interactive multiscale exploration. Through two use cases—navigation and selection—we demonstrate that, as an independent module, our ScaleFree can be directly integrated into diverse pipelines requiring high-performance density estimation in large-scale scientific simulations.

Beyond its technical optimization, our ScaleFree allows us to explore and pursue a novel interaction paradigm for immersive data exploration. Rather than requiring users to learn dedicated 3D navigation techniques—a long-standing challenge due to the complexity and inherent imprecision of 3D interaction—users essentially simply issue commands akin to "*show me that in more detail*." The structure- and context-aware selection paradigm that we paired with the rapid KDE enabled through our ScaleFree approach allows users to simply indicate desired regions of interest to steer their exploration. They can then zoom in to reveal fine-scale detail or adjust their target to focus on another part of the data—ultimately fluidly crossing the scales to find the most informative perspective.

Acknowledgments

Lingyun Yu was partially supported by the National Natural Science Foundation of China, grant № 62272396.

Additional Material Pointers

We share our additional materials (study results/data and analysis scripts, videos) at `osf.io/hfu6e`. We also make our prototypical implementation of the ScaleFree available at the github repository `github.com/LixiangZhao98/ScaleFree`.

References

[1] F. Argelaguet and M. Maignant. GiAnt: Stereoscopic-compliant multi-scale navigation in VEs. In *Proc. VRST*, pp. 269–277. ACM, New York, 2016. doi: 10/p46w 2

[2] F. Bacim, D. Bowman, and M. Pinho. Wayfinding techniques for multiscale virtual environments. In *Proc. 3DUI*, pp. 67–74. IEEE Comp. Soc., Los Alamitos, 2009. doi: 10/fg3dbh 2

[3] A. Backurs, P. Indyk, and T. Wagner. Space and time efficient kernel density estimation in high dimensions. *Adv Neural Inf Process Syst*, 32, art. no. 9251, 10 pages, 2019. url: `papers.nips.cc/paper_files/paper/2019/hash/a2ce8f1706e52936dfad516c23904e3e-Abstract.html`. 3

[4] T. Baguley. Standardized or simple effect size: What should be reported? *Br J Psychol*, 100(3):603–617, 2009. doi: 10/bnw2nb 7

[5] L. Besançon, A. Ynnerman, D. F. Keefe, L. Yu, and T. Isenberg. The state of the art of spatial interfaces for 3D visualization. *Comput Graph Forum*, 40(1):293–326, 2021. doi: 10/gjbpxp 6

[6] R. Brown. HTC vive pro2. Web site: `vr-compare.com/headset/htcvivepro2`. Last accessed: Sep. 2025. 7

[7] C. Brunsdon. Estimating probability surfaces for geographical point data: An adaptive kernel algorithm. *Comput GeoSci*, 21(7):877–894, 1995. doi: 10/dnt6g8 3

[8] A. Bueckle, K. Buehling, P. C. Shih, and K. Börner. 3D virtual reality vs. 2D desktop registration user interface comparison. *PloS one*, 16(10):e0258103, 2021. doi: 10/qmkr 1

[9] J. E. Burt, G. M. Barber, and D. L. Rigby. *Elementary Statistics for Geographers*. Guilford Press, New York, 3rd ed., 2009. doi: 10/br6ngd 3

[10] Z. Cao, J. Han, S. Yang, and X. Jin. Fast best viewpoint selection with geometry-enhanced multiple views and cross-modal distillation. *Vis Comput*, 41(7):5075–5086, 2025. doi: 10/p5p3 7

[11] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proc. CHI*, pp. 181–186. ACM, New York, 1991. doi: 10/cvtdps 6

[12] M. Cautun, R. van de Weygaert, and B. J. Jones. NEXUS: Tracing the cosmic web connection. *Mon Not R Astron Soc*, 429(2):1286–1308, 2013. doi: 10/f4wxgz 1

[13] T. N. Chan, R. Cheng, and M. L. Yiu. QUAD: Quadratic-bound-based kernel density visualization. In *Proc. SIGMOD*, pp. 35–50. ACM, New York, 2020. doi: 10/gkpt7c 3

[14] T. N. Chan, P. L. Ip, B. Zhu, D. Wu, J. Xu, C. S. Jensen et al. Large-scale spatiotemporal kernel density visualization. In *Proc. ICDE*, pp. 99–113. IEEE Comp. Soc., Los Alamitos, 2025. doi: 10/qmw4 3, 9

[15] Y.-C. Chen, S. Ho, P. E. Freeman, C. R. Genovese, and L. Wasserman. Cosmic web reconstruction through density ridges: Method and algorithm. *Mon Not R Astron Soc*, 454(1):1140–1156, 2015. doi: 10/f7wrzq 1

[16] V. Chheang, B. T. Weston, R. W. Cerda, B. Au, B. Giera, P.-T. Bremer et al. A virtual environment for collaborative inspection in additive manufacturing. In *Proc. CHI*, art. no. 26, 7 pages. ACM, New York, 2024. doi: 10/g92v9w 2

[17] I. Cho, J. Li, and Z. Wartell. Multi-scale 7DOF view adjustment. *IEEE Trans Vis Comput Graph*, 24(3):1331–1344, 2018. doi: 10/gcx789 2

[18] G. Cumming. The new statistics: Why and how. *Psychol Sci*, 25(1):7–29, 2014. doi: 10/5k3 7

[19] P. Diggle. A kernel method for smoothing point process data. *J R Stat Soc C (Appl Stat)*, 34(2):138–147, 1985. doi: 10/bsh8ht 1, 3

[20] P. Dragicevic. Fair statistical communication in HCI. In J. Robertson and M. Kaptein, eds., *Modern Statistical Methods for HCI*, chap. 13, pp. 291–330. Springer, Cham, 2016. doi: 10/ggd8gc 7

[21] P. Dragicevic, F. Chevalier, and S. Huot. Running an HCI experiment in multiple parallel universes. In *CHI Extended Abstracts*, pp. 607–618. ACM, New York, 2014. doi: 10/gpcs8n 7

[22] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory Probab Appl*, 14(1):153–158, 1969. doi: 10/bjdbsc 3

[23] B. Ferdosi, H. Buddelmeijer, S. Trager, M. Wilkinson, and J. Roerdink. Comparison of density estimation methods for astronomical datasets. *Astron Astrophys*, 531, art. no. A114, 16 pages, 2011. doi: 10/cj3s5m 3

[24] C. H. Fleming and J. M. Calabrese. A new kernel density estimator for accurate home-range and species-range area estimation. *Methods Ecol Evol*, 8(5):571–579, 2017. doi: 10/f98k6z 3

[25] C.-W. Fu, W.-B. Goh, and J. A. Ng. Multi-touch techniques for exploring large-scale 3D astrophysical simulations. In *Proc. CHI*, pp. 2213–2222. ACM, New York, 2010. doi: 10/drxzf7 2

[26] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. *IEEE Trans Vis Comput Graph*, 11(4):457–468, 2005. doi: 10/cgmnfg 3

[27] R. J. García-Hernández, C. Anthes, M. Wiedemann, and D. Kranzlmüller. Perspectives for using virtual reality to extend visual data mining in information visualization. In *Proc. Aerospace Conf.*, pp. 1151–1161. IEEE Comp. Soc., Los Alamitos, 2016. doi: 10/gtmkhw 1

[28] O. Hahn, C. Porciani, C. M. Carollo, and A. Dekel. Properties of dark matter haloes in clusters, filaments, sheets and voids. *Mon Not R Astron Soc*, 375(2):489–499, 2007. doi: 10/dxbdjj 1, 2

[29] S. Halladjian, D. Kouřil, H. Miao, M. E. Gröller, I. Viola, and T. Isenberg. Multiscale unfolding: Illustratively visualizing the whole genome at a glance. *IEEE Trans Vis Comput Graph*, 28(10):3456–3470, 2022. doi: 10/kt3m 2

[30] S. Halladjian, H. Miao, D. Kouřil, M. E. Gröller, I. Viola, and T. Isenberg. Scale trotter: Illustrative visual travels across negative scales. *IEEE Trans Vis Comput Graph*, 26(1):654–664, 2020. doi: 10/kt3k 2

[31] S. Hart. Nasa-task load index (NASA-TLX); 20 years later. *Proc Hum Factors Ergon Soc Annu Meet*, 50(9):904–908, 2006. doi: 10/fzvtd4 7

[32] B. He, N. K. Govindaraju, Q. Luo, and B. Smith. Efficient gather and scatter operations on graphics processors. In *Proc. SC*, art. no. 46, 12 pages. ACM, New York, 2007. doi: 10/fgtjzk 4

[33] W.-H. Hsu, Y. Zhang, and K.-L. Ma. A multi-criteria approach to camera motion design for volume data animation. *IEEE Trans Vis Comput Graph*, 19(12):2792–2801, 2013. doi: 10/f5h3m3 6

[34] H. M. Kakde. Range searching using KD tree. Technical report, Dept. of Computer Science, Florida State Univ., 2005. Available: `https://users.cs.utah.edu/%7Elifeifei/cis5930/kdtree.pdf`. 4

[35] P. Klacansky, H. Miao, A. Gyulassy, A. Townsend, K. Champley, J. Tringe et al. Virtual inspection of additively manufactured parts. In *Proc. PacificVis*, pp. 81–90. IEEE Comp. Soc., Los Alamitos, 2022. doi: 10/g92v9x 2

[36] R. Kopper, T. Ni, D. A. Bowman, and M. Pinho. Design and evaluation of navigation techniques for multiscale virtual environments. In *Proc. VR*, pp. 175–182. IEEE Comp. Soc., Los Alamitos, 2006. doi: 10/cjs6g6 2

[37] D. Kouřil, T. Isenberg, B. Kozlíková, M. Meyer, M. E. Gröller, and I. Viola. HyperLabels: Browsing of dense and hierarchical molecular 3D models. *IEEE Trans Vis Comput Graph*, 27(8):3493–3504, 2021. doi: 10/kt4m 2

[38] B. Laha, K. Sensharma, J. D. Schiffbauer, and D. A. Bowman. Effects of immersion on visual analysis of volume data. *IEEE Trans Vis Comput Graph*, 18(4):597–606, 2012. doi: 10/fx9b5g 1

[39] O. D. Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *Proc. Pacific Vis*, pp. 171–178. IEEE Comp. Soc., Los Alamitos, 2011. doi: 10/c866v9 3

[40] J. J. LaViola, D. A. Feliz, D. F. Keefe, and R. C. Zeleznik. Hands-free multi-scale navigation in virtual environments. In *Proc. I3D*, pp. 9–15. ACM, New York, 2001. doi: 10/bdmkfc 2

[41] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Comput Graph*, 21(4):163–169, 1987. doi: 10/ft9gsh 6

[42] K. Olsen, R. M. H. Lindrup, and M. Mørup. Think global, adapt local: Learning locally adaptive k-nearest neighbor kernel density estimators. *PMLR*, 238:4114–4122, 2024. url: `proceedings.mlr.press/v238/olsen24a.html`. 9

[43] L. Pavanatto, A. Giovannelli, B. Giera, T. Bremer, H. Miao, and D. A. Bowman. Exploring multiscale navigation of homogeneous and dense objects with progressive refinement in virtual reality. In *Proc. VR*, pp. 228–237. IEEE Comp. Soc., Los Alamitos, 2025. doi: 10/p5fw 2

[44] M.-T. Pelz, M. Schartau, C. J. Somes, V. Lampe, and T. Slawig. A diffusion-based kernel density estimator (diffKDE, version 1) with optimal bandwidth approximation for the analysis of data in geoscience and ecological research. *GeoSci Model Dev*, 16(22):6609–6634, 2023. doi: 10/p5f3 3

[45] S. Pfeifer, N. I. Libeskind, Y. Hoffman, W. A. Hellwing, M. Bilicki, and K. Naidoo. COWS: A filament finder for hessian cosmic web identifiers. *Mon Not R Astron Soc*, 514(1):470–479, 2022. doi: 10/p5fx 1

[46] J. Pivovar, J. DeGuzman, and E. S. Rosenberg. Virtual reality on a SWIM: Scalable world in miniature. In *Proc. VRW*, pp. 912–913. IEEE Comp. Soc., Los Alamitos, 2022. doi: 10/gt2485 2

[47] A. Prouzeau, M. Cordeil, C. Robin, B. Ens, B. H. Thomas, and T. Dwyer. Scaptics and highlight-planes: Immersive interaction techniques for finding occluded features in 3D scatterplots. In *Proc. CHI*, art. no. 325, 12 pages. ACM, New York, 2019. doi: 10/gnkk76 3

[48] K. Saito, M. Yano, H. Hino, T. Shoji, A. Asahara, H. Morita et al. Accelerating small-angle scattering experiments on anisotropic samples using kernel density estimation. *Sci Rep*, 9(1), art. no. 1526, 10 pages, 2019. doi: 10/p5f4 3

[49] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, Hoboken, 2nd ed., 2015. doi: 10/p5f5 3

[50] N. Shivashankar, P. Pranav, V. Natarajan, R. van de Weygaert, E. P. Bos, and S. Rieder. Felix: A topology based framework for visual exploration of cosmic filaments. *IEEE Trans Vis Comput Graph*, 22(6):1745–1759, 2016. doi: 10/f8q5rz 1

[51] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, New York, 1986. doi: 10/gqbn6x 3

[52] D. Song and M. Norman. Looking in, looking out: Exploring multiscale data with virtual reality. *IEEE Comput Sci Eng*, 1(3):53–64, 1994. doi: 10/csw826 2

[53] B. Sousa Santos, P. Dias, A. Pimentel, J.-W. Baggerman, C. Ferreira, S. Silva et al. Head-mounted display versus desktop for 3D navigation in virtual reality: A user study. *Multimedia Tools Appl*, 41(1):161–181, 2009. doi: 10/ccq4dq 1

[54] V. Springel, J. Wang, M. Vogelsberger, A. Ludlow, A. Jenkins, A. Helmi et al. The aquarius project: The subhaloes of galactic haloes. *Mon Not R Astron Soc*, 391(4):1685–1711, 2008. doi: 10/fsjgzw 1, 5, 7, 12

[55] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao et al. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435:629–636, 2005. doi: 10/c3cmxr 1, 2, 5

[56] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM: Interactive worlds in miniature. In *Proc. CHI*, pp. 265–272. ACM, New York, 1995. doi: 10/bx762s 2

[57] G. R. VandenBos, ed. *Publication Manual of the American Psychological Association*. APA, Washington, DC, 6th ed., 2009. url: `apastyle.org/manual`. 7

[58] C. Ware and D. Fleet. Context sensitive flying interface. In *Proc. I3D*, pp. 127–130. ACM, New York, 1997. doi: 10/drbfvq 2

[59] T. Weissker, M. Franzgrote, and T. Kuhlen. Try this for size: Multi-scale teleportation in immersive virtual reality. *IEEE Trans Vis Comput Graph*, 30(5):2298–2308, 2024. doi: 10/p5ft 2

[60] M. Whitlock, S. Smart, and D. A. Szafir. Graphical perception for immersive analytics. In *Proc. VR*, pp. 616–625. IEEE Comp. Soc., Los Alamitos, 2020. doi: 10/gktm78 1

[61] M. H. F. Wilkinson and B. C. Meijer. DATAPLOT: A graphical display package for bacterial morphometry and fluorimetry data. *Comput Methods Programs Biomed*, 47(1):35–49, 1995. doi: 10/dnr53p 3

[62] G. J. Wills. Selection: 524,288 ways to say "this is interesting". In *Proc. InfoVis*, pp. 54–60. IEEE Comp. Soc., Los Alamitos, 1996. doi: 10/bjq6dh 6

[63] C. A. Wingrave, Y. Haciahmetoglu, and D. A. Bowman. Overcoming world in miniature limitations by a scaled and scrolling WIM. In *Proc. 3DUI*, pp. 11–16. IEEE Comp. Soc., Los Alamitos, 2006. doi: 10/bbp9k4 2

[64] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *Vis Comput*, 2(4):227–234, 1986. doi: 10/dndmwc 6

[65] Z. Xie and J. Yan. Kernel density estimation of traffic accidents in a network space. *Comput Env Urban Syst*, 32(5):396–406, 2008. doi: 10/b7gxpq 3

[66] C. Yang, Y. Li, C. Liu, and X. Yuan. Deep learning-based viewpoint recommendation in volume visualization. *J Vis*, 22(5):991–1003, 2019. doi: 10/p5p4 7

[67] Y. Yang, M. Cordeil, J. Beyer, T. Dwyer, K. Marriott, and H. Pfister. Embodied navigation in immersive abstract data visualization: Is overview+detail or zooming better for 3D scatterplots? *IEEE Trans Vis Comput Graph*, 27(2):1214–1224, 2021. doi: 10/ghgt52 6

[68] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input. *IEEE Trans Vis Comput Graph*, 18(12):2245–2254, 2012. doi: 10/f4fv9z 1, 3, 7

[69] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. CAST: Effective and efficient user interaction for context-aware selection in 3D particle clouds. *IEEE Trans Vis Comput Graph*, 22(1):886–895, 2016. doi: 10/kt5n 1, 3, 7

[70] Z. Yuan, M. J. Jarvis, and J. Wang. A flexible method for estimating luminosity functions via kernel density estimation. *Astrophys J Suppl Ser*, 248(1), art. no. 1, 18 pages, 2020. doi: 10/p5qc 3

[71] Z. Yuan, X. Zhang, J. Wang, X. Cheng, and W. Wang. A flexible method for estimating luminosity functions via kernel density estimation. II. Generalization and python implementation. *Astrophys J Suppl Ser*, 260(1), art. no. 10, 15 pages, 2022. doi: 10/p5f2 3

[72] G. Zhang, Q. Huang, A.-X. Zhu, and J. H. Keel. Enabling point pattern analysis on spatial big data using cloud computing: Optimizing and accelerating ripley's k function. *Int J Geogr Inf Sci*, 30(11):2230–2252, 2016. doi: 10/gjkw5j 5

[73] G. Zhang, A.-X. Zhu, and Q. Huang. A GPU-accelerated adaptive kernel density estimation approach for efficient point pattern analysis on spatial big data. *Int J Geogr Inf Sci*, 31(10):2068–2097, 2017. doi: 10/ggfmp5 3, 5

[74] L. Zhao, N. Cao, S. He, H.-N. Liang, and L. Yu. L-wim: Collaborative exploration in immersive environments. In *ISMAR Adjunct Proc.*, pp. 118–123. IEEE Comp. Soc., Los Alamitos, 2022. doi: 10/gtmh3d 1, 2

[75] L. Zhao, T. Isenberg, F. Xie, H.-N. Liang, and L. Yu. MeTACAST: Target-and context-aware spatial selection in VR. *IEEE Trans Vis Comput Graph*, 30(1):480–494, 2024. doi: 10/gtnn25 1, 3, 6, 7

ScaleFree: Dynamic KDE for Multiscale Point Cloud Exploration in VR

Supplementary Materials

We provide additional tables and charts beyond the material that we include in the main paper. For access to the source code, datasets, and analysis scripts used in this work, please refer to `osf.io/hfu6e`. and to github repository `github.com/LixiangZhao98/ScaleFree`.

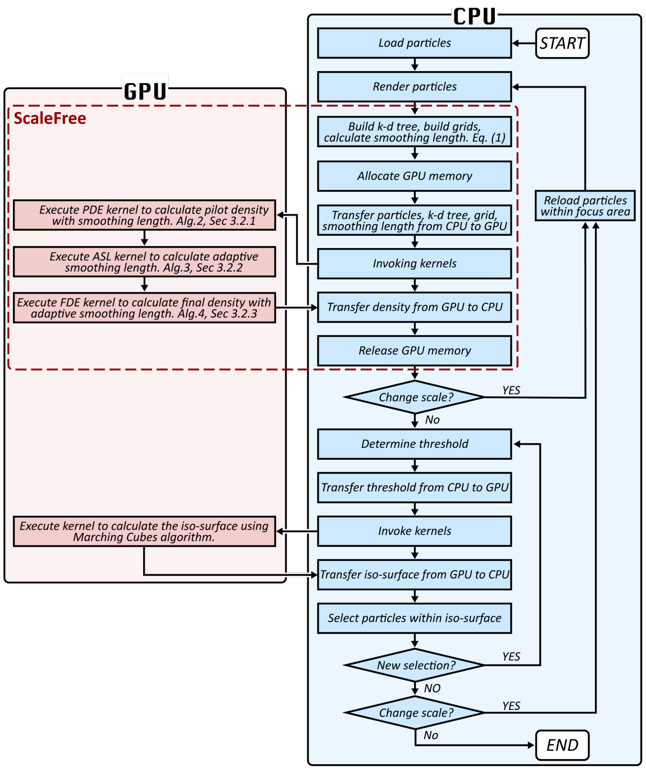

Fig. 9: Workflow of adaptive selection method.

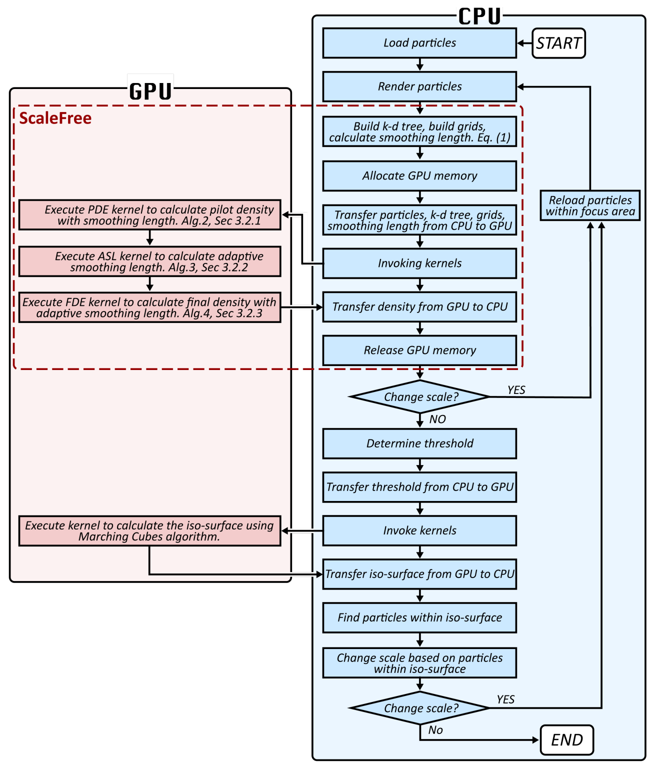

Fig. 10: Workflow of progressive navigation method.

A Workflow illustrations

Fig. 11 shows the flow chart of our GPU-accelerated KDE algorithm described in Sec. 3.2. Fig. 9 shows the flow chart of our scalable selection technique described in Sec. 4.1, while Fig. 10 shows the flow chart of our progressive navigation technique described in Sec. 4.2.

B Additional results from the study

Tab. 2 reports the mean completion times and accuracy scores, whereas Tab. 3 presents the pairwise ratios, as detailed in Sec. 5.3.

C Dataset used in user study

We extracted five timesteps from a cosmological N-body simulation [54] and used them as our datasets. Tasks began at varying scales, with all targets becoming visible as participants zoomed in. Fig. 12 shows the five datasets that were used in the user study described in Sec. 5.

Table 2: The mean task completion times, accuracy scores, and their corresponding 95% confidence intervals.

Technique	Time	CI	FI	CI	MCC	CI
PS	136s	[113,161]	.50	[.44,.55]	.46	[.41,.52]
PM	131s	[113,151]	.64	[.59,.69]	.63	[.58,.68]
DR	100s	[90,112]	.85	[.79,.88]	.83	[.78,.87]

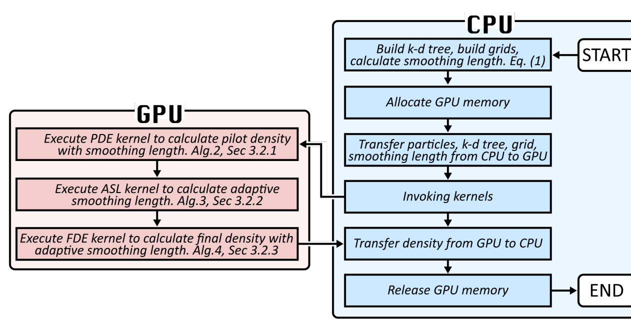

Fig. 11: Workflow of the GPU-accelerated KDE algorithm.

D Parallel programming framework on the GPU

For the GPU parallel programming of our algorithm we follow the HLSL convention. In this model, parallel computation is performed by invoking (programming) kernels, each of which

Table 3: The pairwise ratio of task completion times, accuracy scores, and their corresponding 95% confidence intervals.

Technique	Time	CI	FI	CI	MCC	CI
DR/PM	0.89	[0.70,1.08]	1.33	[1.21,1.44]	1.35	[1.22,1.49]
DR/PS	0.86	[0.66,1.08]	1.74	[1.58,1.92]	1.90	[1.67,2.18]
PM/PS	1.12	[0.87,1.41]	1.36	[1.21,1.53]	1.48	[1.26,1.74]

is executed concurrently by multiple GPU threads. A thread is the basic unit of execution, running the same kernel function independently. We can organize threads into groups defined by `numthreads(tx, ty, tz)` before launching a kernel, which specifies the dimension of threads in the group along the *x*-, *y*-, and *z*-dimensions. Threads within a group share fast group memory and can synchronize their data exchange with group memory via the function `GroupMemoryBarrierWithGroupSync()`. Groups remain independent, thus, threads in different groups cannot access each other's group memory. All GPU threads have access to global memory, which serves as the primary space for data exchange during computation. Before launching kernels, we need to transfer the input data from the CPU to global GPU memory. When launching a compute kernel, `Dispatch(kernel, groupCountX, groupCountY, groupCountZ)` specifies the number of thread groups along the *x*-, *y*-, and *z*-dimensions, thereby defining the overall dispatch layout. The total number of threads executed equals the product of threads per group and the number of groups. During kernel execution, each thread identifies its local (within its group), group, and global (among all threads) indices through semantics such as `SV_GroupIndex`, `SV_GroupID`, and `SV_DispatchThreadID`. Once the kernel completes execution, results are written to GPU buffers and can be returned to the CPU for further processing.

E INTERVIEW QUESTIONS

During the interview phase of our user study, we asked our participants the following questions:

Q1. Perceived delay during interaction. Did you perceive any noticeable delays, lags, or interruptions during scale changes or while making selections when using each technique (PS, PM, and DR)? If yes, please describe when the delay occurred (e.g., during scale transitions, selection, or navigation) and how noticeable it was.

Q2. Impact of delay on interaction experience. If you noticed any delays or performance slowdowns, did they affect your ability to navigate or select regions effectively? Please explain your reasoning.

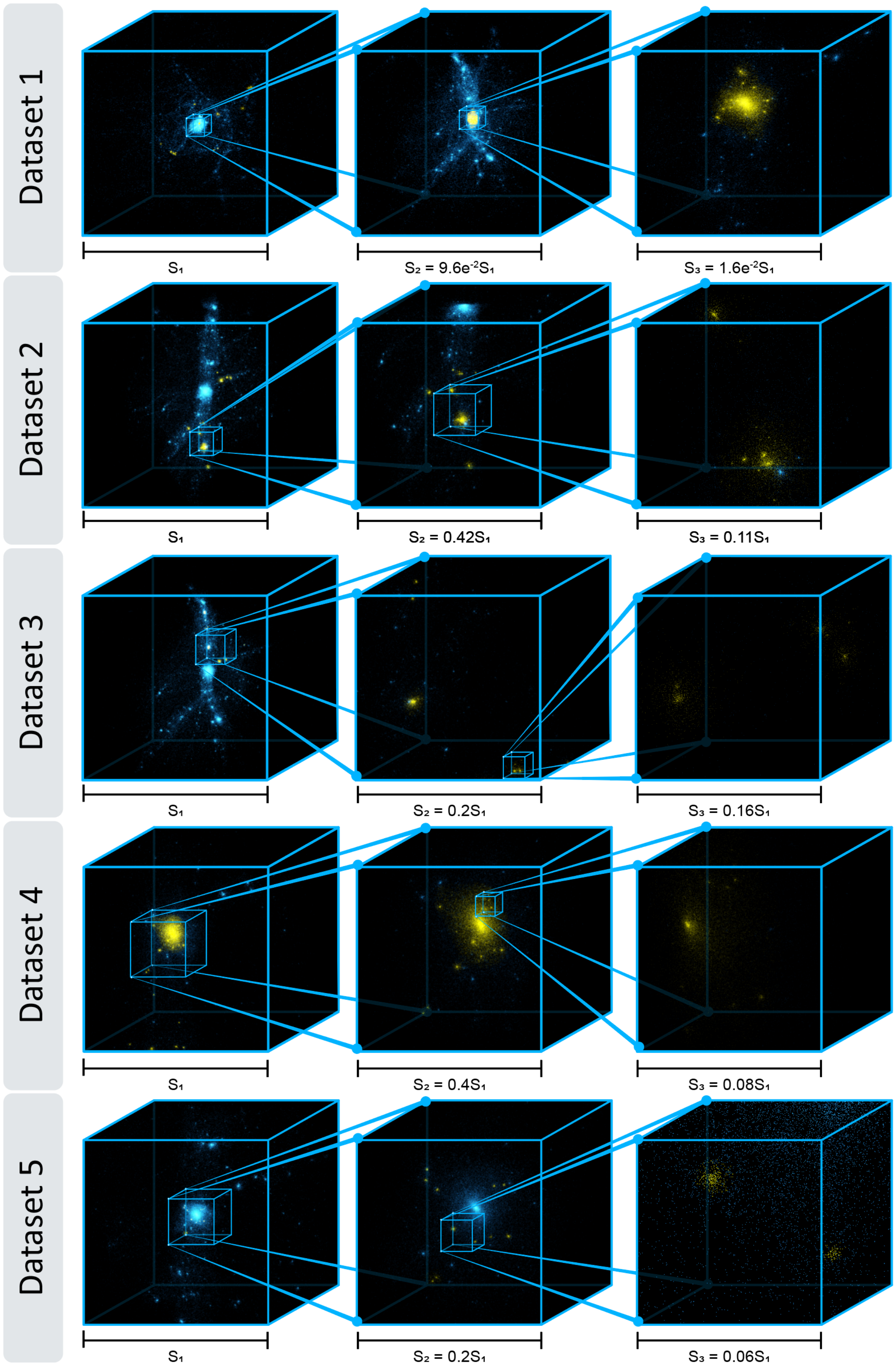

Fig. 12: Experiment datasets.