Distortion-aware Brushing for Reliable Cluster Analysis in Multidimensional Projections

Hyeon Jeon, Michaël Aupetit, Soohyun Lee, Kwon Ko, Youngtaek Kim, Ghulam Jilani Quadri, and Jinwook Seo

Abstract—Brushing is a typical interaction methodology in 2D scatterplots, allowing users to select clustered points within a continuous, enclosed region for further analysis or filtering. However, applying conventional brushing to 2D representations of multidimensional (MD) data, i.e., Multidimensional Projections (MDPs), can lead to unreliable cluster analysis. This unreliability stems from distortions that MDPs introduce, which makes them inaccurately represent the cluster structure of the original MD data. To alleviate this problem, we introduce a novel brushing technique for MDPs called *Distortion-aware brushing*. While users perform brushing, Distortion-aware brushing correct distortions around the currently brushed points by dynamically relocating points in the projection. Data points close to the brushed points in the MD space are brought closer to the brushed points in the 2D projection, while those further away are repelled. Users can thus brush MD clusters more accurately, conducting more reliable cluster analysis. Our user studies with 24 participants show that Distortion-aware brushing significantly outperforms previous brushing techniques for MDPs in accurately separating clusters in the MD space and is robust against distortions. We also present a use case demonstrating the effectiveness of our technique in conducting cluster analysis to solve a real-world analytic problem.

Index Terms—Multidimensional Projections, Distortion-aware Brushing, Brushing, Distortions, Visual Clustering, Cluster Analysis

1 Introduction

Brushing is a process of selecting data points within a continuous region in the 2D space via direct manipulation such as dragging, clicking, or lassoing [1]. This interaction technique allows users to focus on the selected points by labeling or highlighting them [2], [3]. Since its initial introduction [4], brushing has become a common interaction method in visual analytics. One important use of brushing is the identification and analysis of *clusters* within multidimensional (MD) data through multidimensional 2D projections (MDPs) [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. MDPs are often created through dimensionality reduction algorithms like t-SNE [16] and UMAP [17], or by mapping two attributes onto the x and y axes (i.e., orthogonal projections).

However, visual analytics of MD data by applying conventional brushing techniques on MDPs can easily be *unreliable*, i.e., insights gained from the analyses may not accurately reflect the underlying data. Conventional 2D brushing methods typically struggle to detect clusters in the original MD space because MDPs distort the original data [18], [19], [20], [14] (Fig. 1a). For instance, data close in the MD space can be split apart in the 2D layout, forming missing neighbors (MN), while nearby points in the layout can come from remote regions in the MD data space, generating false neighbors (FN). These distortions can stem from various factors, such as complex structure and high dimensionality [21], inappropriate hyperparameter selection, and inappropriate design of DR technique [22], [23] or quality metrics [24], [25]. As a result, conventional brushing techniques might capture 2D clusters that are less cohesive or incomplete when mapped back to their original MD context.

To address this issue, several MDP brushing techniques [26], [27], [28], [6] have been proposed, yet they still face challenges with MDP distortions. These techniques generally work by first brushing a specific 2D region and automatically mapping this selection to an MD region. Depending on the brushing technique employed, the set of brushed points is determined as either the union [26], [27], [28] or the intersection [6] of the sets of points contained within the two regions. This workflow makes the final MD brushing vulnerable to distortions as it depends on a continuous 2D region that is subject to these distortions (Fig. 1a). These techniques may further constrain data analysis by using fixed shapes for the brushed regions, such as circles and hyperspheres [6], or rectangles and hypercubes [27], [28], which cannot effectively capture clusters with non-trivial shapes in real-world datasets.

We propose *Distortion-aware brushing*, a novel brushing technique designed to overcome these issues, enabling users to more accurately identify MD clusters from their 2D projections compared to existing techniques. Our approach addresses distortions in MDPs by persistently drawing points close in the MD space towards the brushed points and repelling those that are farther apart (Fig. 1b). This relocation ensures that 2D brushes accurately mirror the composition of MD clusters in 2D space. Therefore, our technique ensures more reliable cluster analysis of MD data even in cases where MDPs suffer from severe distortions.

Through quantitative studies with 24 participants, we confirm the ability of Distortion-aware brushing to accurately brush

- *Hyeon Jeon, Soohyun Lee, and Jinwook Seo are with Seoul National University, Seoul, Korea. E-mail: {hj, shlee}@hcil.snu.ac.kr, jseo@snu.ac.kr*
- *Michaël Aupetit is with Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar. E-mail: maupetit@hbku.edu.qa*
- *Kwon Ko is an independent researcher. E-mail: hyungkwonko@gmail.com*
- *Youngtaek Kim is with Samsung Electronics, Seoul, Korea. E-mail: ytaek.kim@hcil.snu.ac.kr*
- *Ghulam Jilani Quadri is with the University of Oklahoma. E-mail: quadri@ou.edu*
- *Jinwook Seo and Michaël Aupetit are corresponding authors.*

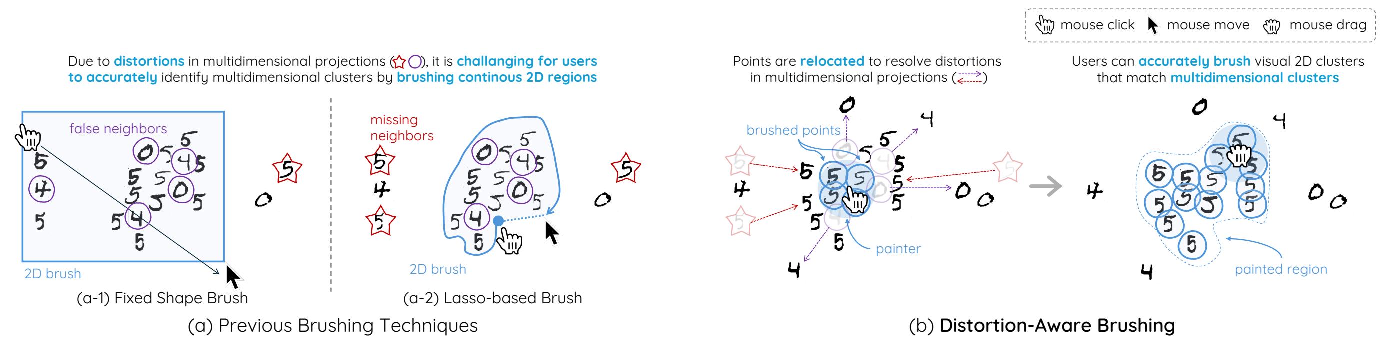
mouse click
mouse move
mouse drag
Due to distortions in multidimensional projections (), it is challenging for users to accurately identify multidimensional clusters by brushing continous 2D regions
false neighbors
2D brush
missing neighbors
2D brush
(a-1) Fixed Shape Brush
(a-2) Lasso-based Brush
(a) Previous Brushing Techniques
Points are relocated to resolve distortions in multidimensional projections ()
brushed points
painter
Users can accurately brush visual 2D clusters that match multidimensional clusters
painted region
(b) Distortion-Aware Brushing

Fig. 1. Comparison between existing brushing techniques (Sect. 2.3) and Distortion-aware brushing in identifying clusters within multidimensional (MD) data through its 2D projection. (a) Previous brushing techniques work by defining a continuous 2D region via direct manipulation (e.g., lassoing). As projections may not accurately reflect original MD data distribution due to distortions, users cannot precisely identify MD clusters using these techniques. (b) In contrast, Distortion-aware brushing supports users to precisely extract MD clusters by resolving distortions based on point relocation, contributing to achieving more reliable MD data analysis.

MD clusters despite distortions, surpassing previous brushing techniques for MDPs. We also showcase how Distortion-aware brushing can be leveraged in practice to solve practical analytic problems. We conclude the paper by discussing the benefits and limitations of Distortion-aware brushing, as well as the potential use cases beyond MD cluster analysis.

2 Background and Related Work

Our work is relevant to three areas: MDP distortions, interactive point relocation, and brushing techniques for MDPs.

2.1 Distortions in Multidimensional Projections

MDPs aim to represent MD data in 2D space while preserving the original characteristics of the given data. For example, dimensionality reduction techniques are used to generate MDPs, providing a visual density-based summary of the data distribution and patterns [14]. However, MDP distortions [14], [18] can interfere with users' ability to analyze clusters or detect outliers in MD data [29], [30], [31], resulting in unreliable visual analytics.

MN (Missing Neighbors) and FN (False Neighbors) [20], [32], [19] are typical MDP distortions that affect the 2D representation of MD data patterns [14] (Fig. 1a). Quantitative metrics such as Trustworthiness and Continuity (T&C) [33] are commonly used in practice [34], [35], [36], [37] to measure and visualize the amount of MN and FN distortions.

MDP can be enriched to visualize the amount and type of the distortions [14] by coloring points [38] or a region around them using heatmaps [38], [39], Voronoi cells [19], [18], [40], or using a network overlay [38], [41]. Although these approaches help explore MDP distortions and reliably analyze the structure of MD data, they are only visual indicators that can inform the brushing process but do not feature brushing actions.

At last, Class-constrained t-SNE [42] and ClassNeRV [43] consider class labels to guide MDP layout for better clustering data. However, they rely on label information that may not correspond to actual MD clusters [24], [44], [45].

2.2 Interactive Points Relocation

Interactive point relocation is widely adopted to explore the underlying structure of MD data. Dust-and-Magnet [46], and iPCA [47] allow interactive steering of the MDP layout based on the attribute values. Another approach is to visualize MD data as snippet images and allow users to arrange MDPs to form groups by visual similarity between the snippets [48], [49]. Yet another technique [50] proposes to interactively brush a cluster of points detected within one MDP layout, freeze it in position, and keep visualizing the remaining data in the same layout with another MDP technique. However, these approaches focus on finding interesting visual cluster patterns under the MDP constraints rather than preventing distortions; thus, they can used to identify interesting insights from MD data but cannot guarantee reliable cluster analysis.

Meanwhile, some previous works aimed to resolve errors locally through relocation. For example, Probing Projections [51] transiently relocates points based on their MD similarity with a user-selected point so that it removes entire MN and FN distortions, but only for the selected point. Proxilens [52] focuses on true MD neighbors of the selected point by pushing FN to the border of a 2D magic lens centered on that point while highlighting MN with proximity coloring [18]. In contrast, instead of transiently resolving distortions around a single point, our technique maintains the correction of distortions related to a set of points, generating a persistent visual pattern (Fig. 1b) that enables the accurate identification and analysis of MD clusters.

2.3 Brushing Multidimensional Projections

We review brushing techniques for MDPs in the literature and categorize them into two groups based on who determines the range of the MD brushed region: users manually adjust the MD region in *axis-guided brushing* while a machine automatically constructs the MD region based on the 2D brushed region in *data-guided brushing*. We provide detailed illustrations of these techniques in Fig. 2.

2.3.1 Axis-Guided Brushing

The early works on brushing MDP were designed to brush along the axes (Fig. 2b). In PRIM-9 [4], brushing is done by adjusting the range of the 2D brush region along two axes of an orthogonal projection, forming a rectangular region. Becker et al. [2], [3] applied the same strategy to a Scatterplot Matrix (SPLOM) consisting of multiple linked orthogonal projections, showing how brushed points in one projection are clustered in the other ones. However, these techniques allow for the brushing of at most two axes, making it challenging to discover MD structures that span more dimensions. To resolve this problem, Ward proposed N-dimensional brushing as a feature of XmdvTool [27] (Fig. 2c),

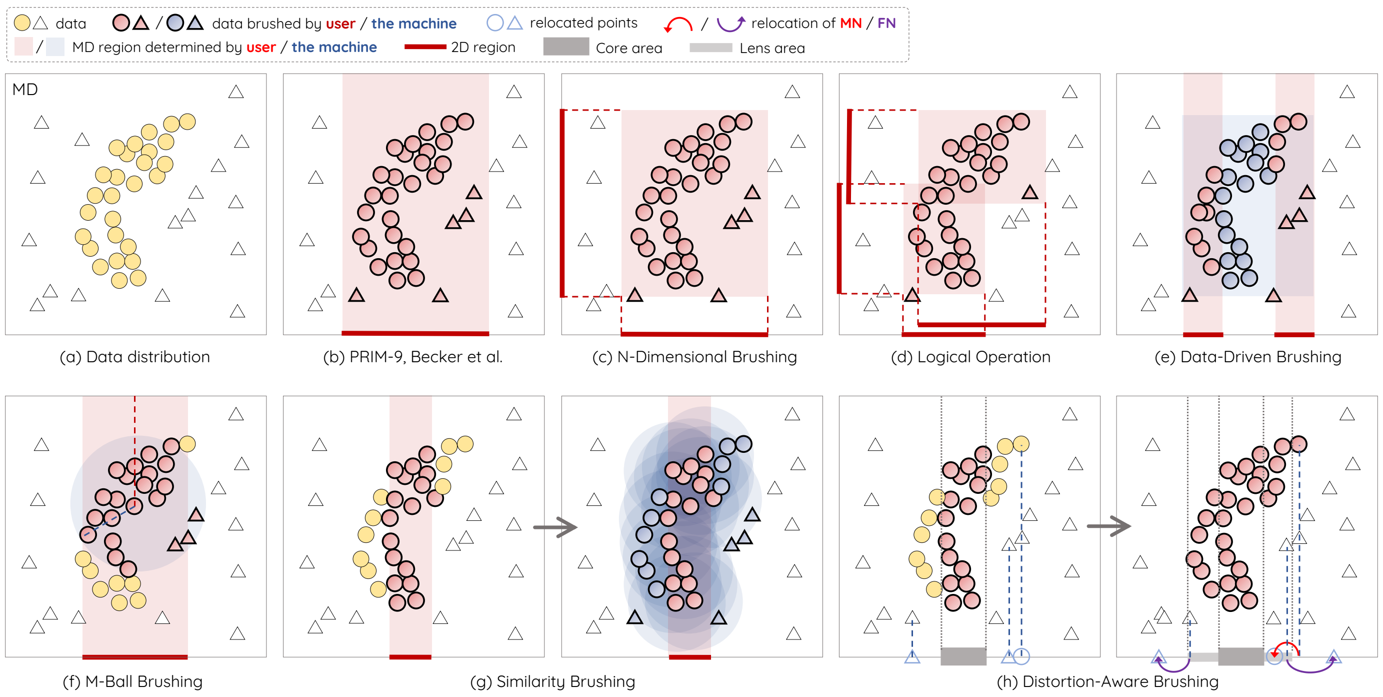

Fig. 2. Previous brushing techniques for MDPs (Section 2.2) (a–g) and Distortion-aware brushing (h). The MD space is depicted as a 2D scatterplot in all views, and the projection is assumed to be a 1D orthogonal projection on the x-axis (b,e,f,g) or the x and y axes (c,d). (a) We assume users seek to brush all MD data forming a cluster (yellow circles) except surrounding outliers (white triangles). In previous techniques (b–g), 2D brushed regions are represented as one or several 1D lines (bold red lines), and MD regions are determined by users and the machine is depicted as red and blue shaded areas, respectively; brushes are defined as compact regions in 2D and MD spaces, and the system reports brushed points as an intersection or a union of the points enclosed therein. In contrast, in Distortion-aware brushing (h), the technique only manages the discrete brushed sets of points and a 2D lens (plain grey lines) with the core area (thick) and lens area (thin). Points are relocated in or out of the lens area (red and purple arrows), reflecting data distribution near the brushed points in the MD space, correcting for MN and FN distortions relative to the 2D lens. Users can readily brush the MD cluster continuously, resolving FN and MN on the fly.

allowing users to define multiple 2D brushes within different projections of a SPLOM. Therefore, users can adjust ranges along more than two dimensions, generating an MD box enclosing data. A later version of XmdvTool [28] enables users to apply logical operators (e.g., AND, XOR) between multiple MD regions for more flexibility (Fig. 2d).

Still, these SPLOM-based techniques have difficulty selecting clusters in the MD space when clusters span more than two dimensions, as the techniques inherently rely on multiple 2D orthogonal projections, each subject to FN distortions. Furthermore, SPLOM uses $O(M^2)$ scatterplots for representing M-dimensional data, making brushing hardly practical when M is large. Parallel coordinate plots (PCP) [53] provide an alternative axis-based representation of MD data denser than SPLOM, for which advanced techniques for brushing have been proposed [54], [55]. However, PCP shatters the MD clusters into M-linked 1D projections (axes), each with more FN distortions than with 2D projections. Axis-guided brushing also does not generate a compact MD region if used with nonlinear dimensionality reduction techniques such as t-SNE or UMAP because the projection space axes are neither linear nor continuous mappings from the M original dimensions.

2.3.2 Data-Guided Brushing

Data-guided approaches were proposed to overcome the problems of axis-guided brushing. The techniques follow a typical workflow: (1) users determine the 2D region through interaction (e.g., painting); (2) a machine automatically constructs the MD region based on the user-defined 2D region; (3) the brushed points are defined as a union or intersection of the set of points within MD and 2D regions.

For example, Data-driven brushing [28] (Fig. 2e) allows users to define a 2D region by generating a box that encloses certain areas in the projection, which then generates an MD region as a minimum-size M-cube enclosing all the data corresponding to the painted points. In M-ball brushing [6] (Fig. 2f), users can capture MD clusters by defining a circular 2D region; the system then automatically formulates a M-ball MD region covering the corresponding data in the MD space. In both approaches, the MD region is bound to convex shapes, making it hard to discover non-trivial (i.e., any-shaped) clusters. In the Similarity brushing proposed by Novotný and Hauser [26] (Fig. 2g), when users paint a visual cluster as the 2D region, the MD region is defined as the area covered by the union of M-balls with radius δ centered on the MD data corresponding to the 2D painted points. The shape of the brushed cluster is determined by the points covered by the 2D brush, so it is not bound to convex-shaped regions.

Still, all these MD brushing techniques are vulnerable to MDP distortions. Indeed, if the 2D region contains FN, the painted data might belong to more than one cluster in the MD space. Moreover, an MD cluster can be split in the projection due to MN, so users will have to brush each of these 2D clusters separately, or even worse, will ignore them if points are spread apart, not forming clear 2D clusters.

Distortion-aware brushing (Fig. 2h) is a data-guided technique that resolves these issues through continuous point relocation. Instead of keeping continuous 2D and MD regions, we only consider the brushed data points. MN and FN are resolved by pulling them close to or pushing them apart from the currently brushed 2D points, respectively. In contrast to other techniques, point relocation always generates 2D visual clusters that match MD ones, thus faithfully representing users' mental model. Therefore, the method supports a more reliable control of the interactive MD cluster analysis process.

3 Design Objectives

Our design objectives tackle the drawbacks of previous brushing techniques for MDPs (O1 O2 O3) while maintaining their strengths (O4) (Sect. 2.3.2).

O1 *Guide brushing by visually reflecting MD clusters*

Previous data-guided brushing techniques work by converting a 2D brushed region into an MD region. However, the inconceivability of the MD space makes it difficult for users to understand this conversion, thus lowering the interpretability and controllability of these techniques. Instead, Distortion-aware brushing performs the conversion in the opposite direction: *2D points are relocated to form a visual cluster that reflects an MD cluster*.

O2 *Allow brushing to be robust against any MDP distortions*

Previous brushing approaches lead to unreliable cluster analysis as they rely on a compact 2D projection region, which is vulnerable to MDP distortions. In contrast, Distortion-aware brushing continuously relocates points to *ensure that 2D neighbors are always true MD neighbors and the MD cluster under focus is not split in the projection*, making the technique work robustly regardless of the type and the amount of distortions.

O3 *Allow accurate brushing of non-trivial-shaped MD clusters*

In most previous brushing techniques [28], [6], [27], the MD region's shape enclosing brushed points is limited to regular compact domains (hyperspheres or hypercubes). This limitation makes the techniques hardly support the discovery of clusters with non-trivial shapes typical of real-world MD data. For example, fitting such a fixed-shaped brush to a non-trivial-shaped cluster can capture out-of-cluster points. On the other hand, reducing the brush size to avoid capturing out-of-cluster points may result in missing in-cluster ones, lowering the clustering accuracy. In contrast, Distortion-aware brushing manages *a discrete set of brushed points instead of compact 2D and MD regions*. This enables users to gradually append new points to the 2D brush corresponding to true MD neighbors of the already brushed points, facilitating the discovery of clusters with arbitrary non-trivial shapes.

O4 *Minimize the number of hand-tuned hyperparameters*

Previous brushing techniques have at most one hyperparameter that affects brushing results, making them easy to use and learn. *M*-Ball Brushing [6] even dynamically adjusts hyperparameter values based on the status of the 2D brush, further reducing the burden of using the technique. Similarly, we design Distortion-aware brushing to have *a single hyperparameter that affects the granularity of the brushed clusters* and make its value *controlled automatically to reflect the current brushing status*.

4 Distortion-aware brushing

Distortion-aware brushing relocates points within and around the current brushed points in the projection by faithfully reflecting the data distribution around the brushed points in the MD space (O1 O2). The set of brushed points grows progressively to form a visual cluster that accurately reflects the MD cluster (O3). By doing so, Distortion-aware brushing enables users to conduct a more reliable detailed analysis of MD clusters.

In this section, we first describe the design of Distortion-aware brushing following the overall workflow of the technique (Sect. 4.1). We then describe additional features of the technique developed for its practical usage in visual analytics (Sect. 4.2). We describe our technical details in Appendix H.

4.1 Workflow

Distortion-aware brushing follows a four-step workflow (Fig. 3):

Step 1 : Users inspect how the MD data distribution matches the 2D visual cluster patterns to decide the best places to initiate brushing.

Step 2 : Users inspect local MN and FN distortions around these candidate places by hovering the green disc-shaped painter over the points within these places.

Step 3 : A pause of mouse move initiates a transient relocation of the points, repulsing FN and attracting MN, correcting the local distortions to inform the brushing decision. Moving the mouse again reverses the relocation and returns to Step 2 .

Step 4 : Users execute brushing by dragging the painter while the mouse button is pressed, progressively capturing the covered points. Lens construction and point relocation are performed iteratively based on the current 2D location of brushed points and the MD distribution in their vicinity.

Step 1 *Inspecting global distortions*

For reliable MD cluster analysis, brushing shall ideally initiate at a place close to the core of an actual MD cluster. To support the task, we encode each point as a snippet representing the corresponding MD datum (Fig. 1). For example, each point in image datasets can be represented as an image snippet. Also, points in tabular datasets can be represented using glyphs [56], e.g., aster plots [57]. By visualizing snippets, users can spot 2D visual clusters (high mutual proximity) matching with MD clusters (high similarity between snippets) (O1 O2), recognizing these locations as good candidates for initiating brushing.

We further aid this step by encoding the MD density of the data in the MD space through the opacity of the corresponding points or snippets (Fig. 3 (a)). Based on density encoding, users can check the trustworthiness of a 2D visual cluster by comparing whether the 2D density, represented by proximity between points, matches MD density. A visual cluster with higher 2D density than other visual clusters, containing points with higher MD density than others, can be considered a good candidate. The best location to start the brushing is the high-density central part of the 2D visual cluster, which corresponds to the core region of the MD cluster. Other cases showing a density mismatch reflect MDP distortions and should be avoided. More comprehensive distortion visualizations could be used (e.g., visualizing FN and MN distortions [19], [36]), but we preferred visualizing the density to make users easily learn the technique.

MD density. We define the MD density of a point p as $\texttt{dens}(p) = \sum_{q \in P} \texttt{sim}_k(p,q)$, borrowing the definition of Density-peak

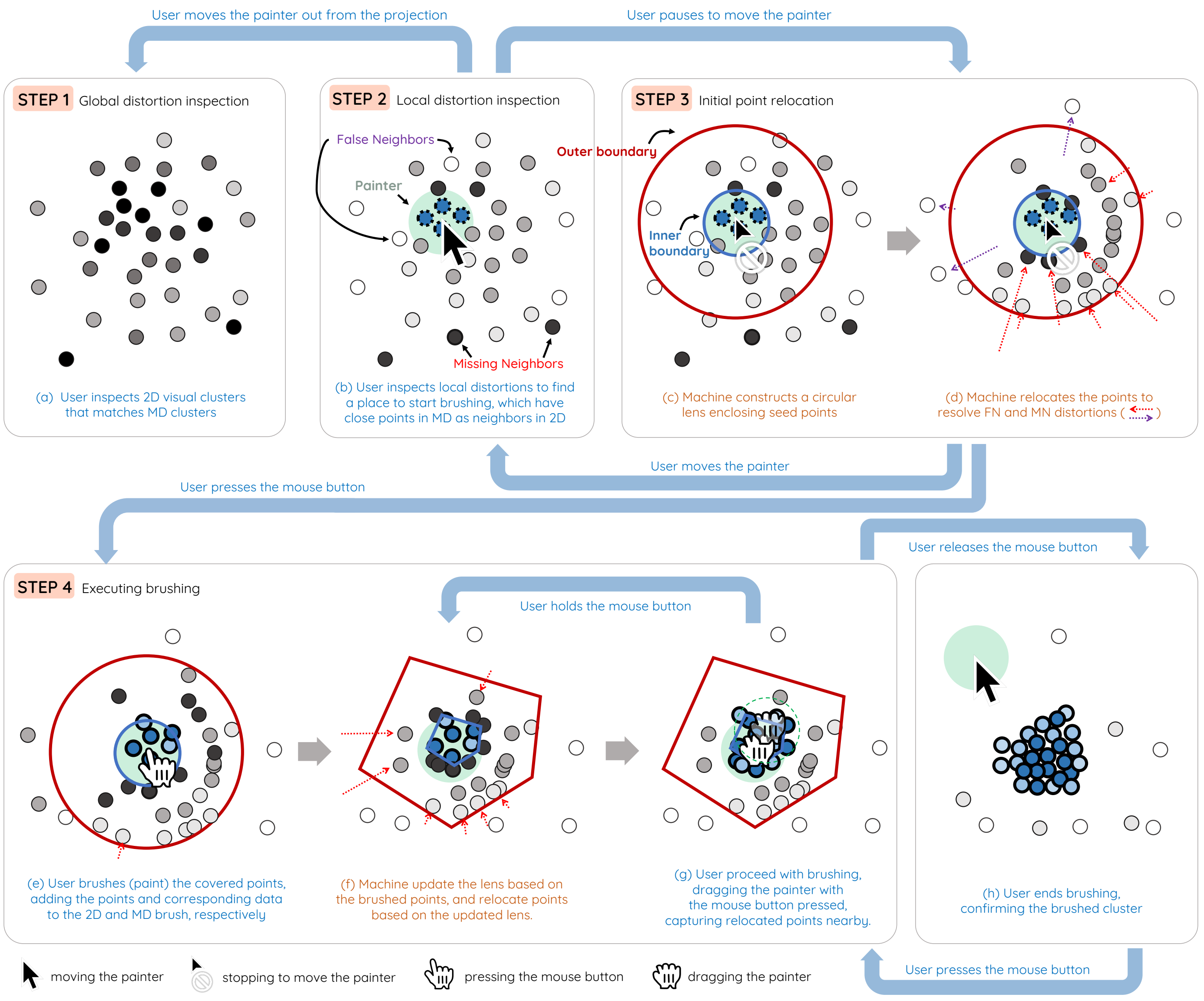
User moves the painter out from the projection
User pauses to move the painter
STEP 1 Global distortion inspection
STEP 2 Local distortion inspection
STEP 3 Initial point relocation
False Neighbors
Painter
Outer boundary
Inner boundary
Missing Neighbors
(a) User inspects 2D visual clusters that matches MD clusters
(b) User inspects local distortions to find a place to start brushing, which have close points in MD as neighbors in 2D
(c) Machine constructs a circular lens enclosing seed points
(d) Machine relocates the points to resolve FN and MN distortions
User moves the painter
User presses the mouse button
User releases the mouse button
STEP 4 Executing brushing
User holds the mouse button
(e) User brushes (paint) the covered points, adding the points and corresponding data to the 2D and MD brush, respectively
(f) Machine update the lens based on the brushed points, and relocate points based on the relocated lens.
(g) User proceed with brushing, dragging the painter with the mouse button pressed, capturing relocated points nearby.
(h) User ends brushing, confirming the brushed cluster
moving the painter
stopping to move the painter
pressing the mouse button
dragging the painter
User presses the mouse button

Fig. 3. Overall workflow of Distortion-aware brushing (4.1). The technique features a lens with inner and outer boundaries depicted as bold blue and red closed lines, respectively. Users' actions are explained with blue text and arrows, while the machine's actions are detailed in orange. Data points are represented as small circles, i.e., dots, where seed and brushed points are highlighted using thick dotted and solid borders. Seed and brushed points are also highlighted in blue color. The opacity of data points depicts MD density in Step 1 and represents MD closensss to the seed or brushed points in the following steps.

clustering [58], [59], where $\text{sim}_k(p,q)$ represents the similarity between points p and q, and P denotes the entire set of points in the data.

MD similarity measure. For the similarity measure, we use the Shared-Nearest Neighbors (SNN) similarity [60], which assigns higher similarity to the pairs of points sharing more k-Nearest Neighbors (kNN). Formally, the SNN similarity between p and q is defined as $\text{sim}_k(p,q) = \sum_{(m,n)\in S_{p,q}}(k+1-m)\cdot(k+1-n)$; $S_{p,q}$ represents a set containing pairs (m,n) fulfilling $p_m = q_n$ where p_i denotes an i-th nearest neighbor of p and q_i denotes an i-th nearest neighbor of q. We select SNN as this metric is shift-invariant [22], making it alleviate the curse of dimensionality [22], [44] and thus better represents the cluster structure of high-dimensional spaces compared to other metrics (e.g., kNN, Euclidean distance) [60], [59], [36]. We fix k as the square root of the number of points, following the recommendation by Chaudhuri and Dasgupta [61].

Step 2 *Inspecting local distortions*

In this step, users can "skim" local distortion of the projection by moving the painter over the points (Fig. 3 (b)). This is done by (1) finding *seed points* within a painter, then (2) visualizing the MD closeness of any point to the seed points [18]. Compared to Step 1, this step provides a more stringent inspection of local distortions that can help users locate the best candidate for initiating brushing.

The detailed procedure is as follows. First, the machine determines the seed points as condensed points the painter covers. This is done by identifying the covered point with the highest MD density and only using its close neighbors as seed points (see *Finding seed points* below). It is important to avoid using every point covered by the painter as seed points because they may consist of points coming from two or more distinct MD clusters due to FN; if this happens, new points brushed from these distinct MD clusters will erroneously agglomerate into a single visual

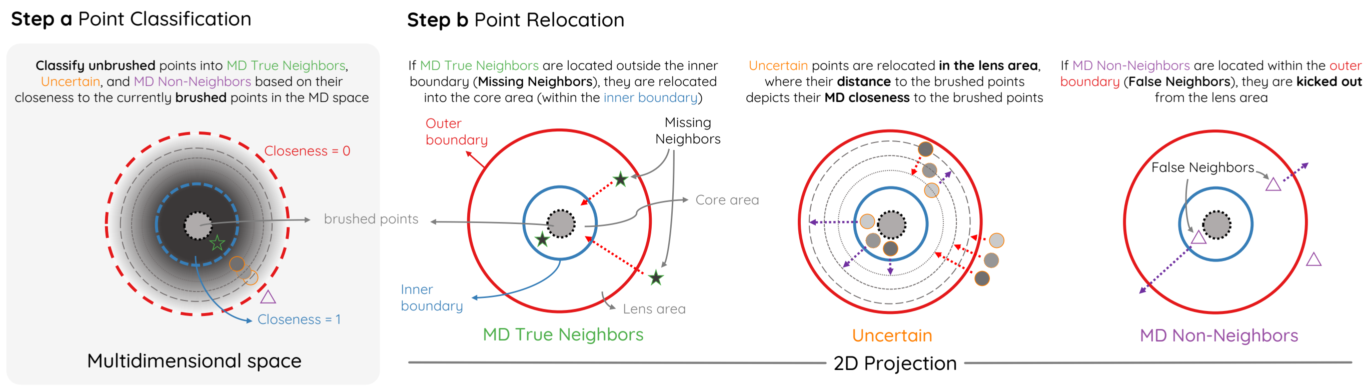
Step a Point Classification
Step b Point Relocation
Classify unbrushed points into MD True Neighbors, Uncertain, and MD Non-Neighbors based on their closeness to the currently brushed points in the MD space
Closeness = 0
brushed points
Closeness = 1
Multidimensional space
If MD True Neighbors are located outside the inner boundary (Missing Neighbors), they are relocated into the core area (within the inner boundary)
Outer boundary
Missing Neighbors
Core area
Inner boundary
Lens area
MD True Neighbors
Uncertain points are relocated in the lens area, where their distance to the brushed points depicts their MD closeness to the brushed points
Uncertain
If MD Non-Neighbors are located within the outer boundary (False Neighbors), they are kicked out from the lens area
False Neighbors
MD Non-Neighbors
2D Projection

Fig. 4. Illustration on how Distortion-aware brushing relocates points in Step 3 and Step 4. The machine first examines the MD closeness of unbrushed points to the brushed points (or seed points in Step 3) (3-a), then relocates those points in the projection to reflect that MD closeness.

cluster (O1 O2). The seed points are then highlighted with a color corresponding to the current brush.

Then, the remaining points' MD closeness to the seed points is encoded as their opacity (i.e., closer points are darker). This graphical encoding informs users' brushing decisions by indicating more reliable locations to start brushing (O1). Users can identify FN in the projection as points with bright or non-highlighted markers near the seed points and MN as points with dark markers far from the seed points.

Finding seed points. Constructing a set of seed points starts by identifying the initial seed point $p_{initial}$ with the highest MD density among the subset of points C covered by the painter: $p_{initial} = \arg\max_{p \in C} \texttt{dens}(p)$. Then, a set of seed points is defined as the κ nearest neighbors (κNN) of $p_{initial}$ based on SNN similarity in the MD space, where the machine automatically sets κ as the maximum value such that all κNN are still covered by the painter (O4). Users can adjust the size of the painter and so κ, to make MD brushing more condensed or relaxed. By this definition, the seed points and the initial brush cannot contain FN.

MD closeness. Closeness between a data p and a set of points C is defined as follows: $\texttt{close}_{\kappa}(p, C) = \sum_{q \in (\kappa NN \cap C)} \texttt{sim}_k(q, p) / \sum_{q \in \kappa NN} \texttt{sim}_k(q, p)$. Intuitively, the more κNN of p are members of the cluster C, the closer p is to C. Moreover, by definition, the seed points of the initial brush have the maximum closeness. We do not naively average the similarity of p to the points within C because this would make the closeness depend on cluster characteristics unrelated to closeness, such as density or size.

Step 3 *Initiating point relocation*

Though global (Step 1) and local distortion (Step 2) inspections support users in finding a good candidate region, the points nearby likely suffer from MDP distortions, not accurately reflecting the local MD data distribution. We thus provide a *point relocation* process that corrects the distortions relative to the current brush (initially, the seed points covered by the painter) (Fig. 3 (c-d)). Inspired by the Proxilens approach [52], users can trigger transient point relocation by halting the painter for a short time, which is determined as 800ms in our implementation through an iterative design process. Correcting for local distortions can be viewed as "jumping" into the MD space, as it makes the 2D distribution around the painter and the currently brushed points better reflect the local MD data distribution. Users can reverse the current point relocation (jumping back to the 2D space) by moving the painter again.

To perform relocation, the system first constructs a magic lens around seed points in the projection (Fig. 3 (c)). The lens consists of (1) a core area delimited by an inner boundary that tightly encloses seed points and (2) an annulus lens area around the core, enclosed between the inner boundary and an outer boundary (Fig. 4, blue and red solid circular boundaries, respectively).

Afterward, point relocation is performed according to the lenses so that points distribution around seed points in the projection can reflect the distribution around the seed points in the MD space(O1) (Fig. 3 (d)). The relocation of a point thus depends on its MD closeness to the seed points (Fig. 4). If the closeness is 1, the point is considered as *MD True Neighbors*, which means that the point belongs to the core MD cluster formed by the seed points. If the closeness is 0, the point is categorized with *MD Non-Neighbors*, denoting that it is far apart from the seed points in the MD space. If the value is between 0 and 1, the point is considered to be *Uncertain*, forming a "fuzzy" neighborhood in the MD space. While the machine relocates *MD True Neighbors* (*i.e* MNs) into the inner boundary, *MD Non-Neighbors* (*i.e.* FNs) are repelled from the lens area outside the outer boundary, and *Uncertain* points are relocated within the lens area, close to the inner boundary, in proportion to their MD closeness to the core MD cluster. Points are relocated to the correct position with an animated transition. Relocation corrects MDP distortions, making points in the inner lens correspond to True Neighbors, as FNs are repelled from the lens and MNs attracted within the lens (O1).

It is worth noting that the *Uncertain* points and their interpolated relocation are a crucial part of Distortion-aware brushing, as it is up to users to decide whether a point will be brushed or not. The *Uncertain* points falling into the outer lens are natural candidates for brushing; they are also geometrically the next ones that can be captured by the painter (O1 O2) (see *Dynamical update* in Step 4).

Initial lens construction. The initial inner and outer lens boundaries are defined as circular boundaries centered on the painter. We set τ as both the inner boundary's radius and the radius of the painter. By doing so, the painter covers both seed points and *MD True Neighbors*, forcing them to be brushed when users execute brushing (Step 4). We also define the radius of the outer boundary as 3τ, setting the lens area's width as the painter's diameter (i.e., 2τ). We justify this decision while describing how we construct the outer lens boundary in Step 4 (*Outer boundary construction*).

Step 4 *Executing brushing*

Once the transient relocation is settled, users can initiate brushing by pressing the mouse button (Fig. 3 (e)). The machine appends the points covered by the painter (which naturally contains seed points) to the brushed set of points. The brushed points are highlighted with the color corresponding to the brush. Then, the machine updates the lens and relocates the remaining points based on the new set of brushed points (Fig. 3 (f)). If users drag the painter while keeping the mouse button pressed (Fig. 3 (g)), the brushed points, the painter, and the lens are updated accordingly, and the relocation takes place again in a continuous cycle. Such gradual updates enable users to agglomerate new brushed points in an arbitrary direction in the MD space, thus allowing the brushing of an MD cluster with an arbitrary shape (O3).

If users decide to end brushing and confirm the brushed MD cluster, they can end the cycle by releasing the mouse button. Such decisions can be made when (1) there are no more unbrushed image snippets (i.e., data points) that look similar to the ones inside the set of brushed snippets or (2) newly brushed points have a relatively lower density than the previously added points. Auxiliary visualizations (e.g., parallel coordinates plot or heatmap) can also guide users in deciding the boundary of brushed clusters (Sect. 6). Here, users can again go back to brushing by pressing the mouse button or erase points that are not intended to be in the brush (Sect. 4.2.1).

During brushing, the brushed points are also relocated to aid interaction. First, we uniformize their locations. The uniformization removes empty space within the inner boundary, making the 2D proximity between each unbrushed point and the brushed points accurately reflect their closeness. It also removes overlap between the points, supporting users in visually investigating snippets. Then, the points are successively relocated to better reflect the original data distribution of the MD cluster. The points in which corresponding data have high MD closeness to the brushed points move near the center of the core area of the lens, and the ones with low closeness move near the boundary of the inner lens. We achieved this by swapping the positions of the points to align their distances to the inner boundary with closeness. This makes the 2D visual cluster a better match with the MD cluster (O1) and helps users readily erase points with low closeness (O2).

Inner boundary construction. While brushing, the inner boundary is set as a convex hull enclosing the brushed points. We used a convex hull as it is computationally inexpensive ($O(n \log n)$ for n points) while tightly enclosing the brushed points compared to alternatives (e.g., boundary circle) and has no hyperparameter to tune (O4). As a result, the brush is always convex, even if the MD cluster has a more convoluted shape.

Outer boundary construction. The outer boundary is constructed by offsetting each corner of the inner boundary to maintain the width of the lens area. Thus, points with the same MD closeness to brushed points are displayed at the same distance to the core lens. This approach also makes the relocation of points isotropic, not biased by the direction from which they originate. It supports our encoding where visual proximity matters while provenance direction has no specific importance. We detail this justification and design alternatives in Appendix D.

As mentioned in Step 3, we use 2τ offset to match the lens area's width with the painter's diameter. Thus, when an *Uncertain* point enters the painter, lying on its circular edge, its MD closeness to the brushed points corresponds roughly to the proportion of the painter area overlapping the core lens, a visual indicator easy to estimate. For example, if the painter fully overlaps with the core lens, any point within the painter can belong to the brush but none within the lens area (i.e., the accepted closeness is only $\alpha = 1$). If the painter α-overlap with the core lens, points with MD closeness above α can belong to the brush (the accepted closeness range is $[\alpha, 1]$). Finally, if the painter stays entirely within the lens area, outside the core lens, touching the outer boundary, all the points covered by the painter can belong to the brush (the accepted closeness range is total: $[0, 1]$).

Dynamical update. As the brush is dynamically updated with the *Uncertain* points captured by the painter, the brush grows up on the painter's side at a certain speed due to relocation animated transitions and **core lens uniformization** (see below). This increases the overlap area of the core lens with the painter, hence the acceptance threshold α, and reduces the painter's covering of the lens area, preventing further *Uncertain* points from being captured and a positive feedback loop that would lead to a loss of control. Users must keep moving the painter toward the nearby lens area to lower the acceptance threshold back to the desired value, capturing new *Uncertain* points down to that level. Thus, users intending to brush MD data carefully will naturally move painter slower, thus maintaining a higher α and a more stringent acceptance of *Uncertain* points as *MD True Neighbors*.

Core lens uniformization. We used the centroidal Voronoi tessellation [62] based on Lloyd's algorithm [63] to uniformize the distribution of brushed points. We clip the Voronoi cells to fit the inner boundary so that the points remain within the core lens.

4.2 Features for Enhanced Usability

We discuss features of Distortion-aware brushing that improve its usability and applicability: the erasing mode, the use of multiple brushes, and their contextualization.

4.2.1 *Erasing Mode*

Users can erase points that are not intended to be brushed using erase mode. The erase mode is enabled when (1) brushing is paused by releasing the mouse button during Step 4 and (2) the keyboard shift button is pressed. When the mode is enabled, users can erase points by hovering the painter over the points while keeping the mouse button pressed. The painter's color becomes red during the erase mode.

Natural candidates for erasing are points with low MD closeness to the brushed points. As the system pushes such points with low closeness near the inner boundary (Step 4), users can readily erase them by moving the painter in the lens area of the brush slightly overlapping the core lens to capture these outliers. The erased points are then relocated like other unbrushed points based on their closeness to the updated brush.

4.2.2 *Multiple Brushes*

In previous brushing techniques, multiple brushes were often provided to compare multiple sets of points, widening the analytic search space [6], [27]. Martin and Ward [27] even proposed to perform logical operations on multiple brushes.

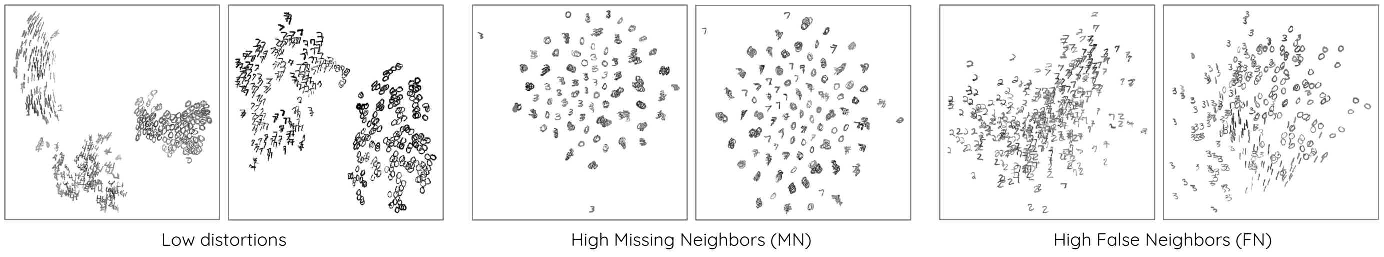

Fig. 5. The example projections (i.e., stimuli) used in our experiments with different amounts of distortions. In study 1 (Sect. 5.1, we treat the amount of distortions as an independent variable, namely DistortionAmount. In study 2 (Sect. 5.2), the amount of distortions is controlled as a confounding variable.

We also allow users to control multiple brushes while distinguishing them with different colors. Users can pause the current brush and switch the focus to another brush by pressing a button with the same brush color. We showcase the utility of multiple brushes in our use case (Sect. 6).

4.2.3 Contextualization within the Original Projection

Point relocation of Distortion-aware brushing resolves distortions around the brushed points (O2). However, the relocation can generate more distortions in other areas of the MDP, making it hard for users to understand the brushed cluster in the context of the original MDP. To alleviate this side effect, we allow users to contextualize all the brushes within the original MDP, relocating every points to their original position with an animated transition while preserving their color to indicate the brush identity.

5 User Studies for Evaluating the Robustness of Distortion-aware brushing

We conduct two controlled user studies to validate the effectiveness of Distortion-aware brushing. The studies investigate how Distortion-aware brushing's performance in extracting MD clusters (accuracy and task completion time) is affected by the amount of distortions (O2; Sect. 5.1) and the non-triviality of the shape of MD clusters (O3; Sect. 5.2). Both studies compare Distortion-aware brushing against three existing brushing techniques for MDP. We do not empirically investigate the satisfaction of O1 and O4, as they are fulfilled by design (Sect. 4.1). The approval for the user studies is granted by Seoul National University IRB.

5.1 User Study 1: Robustness Against Distortions

5.1.1 Objectives and Design

We aim to compare Distortion-aware brushing with baselines regarding accuracy in capturing MD clusters and their robustness to the amount of DR distortions. We also want to assess how global (Step 1) and local (Step 2) distortion inspections (Sect. 4.1) affect user performance of the brushing techniques. To achieve such goals, we design the study to have three independent variables:

- Amount and type of distortions (DistortionAmount)
  - *Low distortion*, *High MN*, and *High FN*
- MD brushing techniques (Techniques)
  - Three baseline techniques (*Data-driven brushing* [28], *M-Ball Brushing* [6], *Similarity brushing* [26]), and *Distortion-aware brushing*.
- Availability of global and local distortion inspections (DistortionInspection)
  - *No inspection*, *Only global*, and *Both global and local*

resulting in a within-subject experiment with 3 [DistortionAmount] $\times$ 4 [Techniques] $\times$ 3 [DistortionInspection] $=$ 36 trials per participant. Note that we select data-guided brushing techniques (Sect. 2.3.2) as they are more accurate than axis-guided brushing (Sect. 2.3.1) and can also be applied beyond orthogonal projections (Refer to Appendix B for the design of baseline techniques). Controlling DistortionInspection also ensures the fairness of our experiment in comparing Distortion-aware brushing and baseline techniques. This is because baseline techniques' task performance may also benefit from the inspection functionalities. Detailed study design is as follows:

Task. We ask participants to perform interactive labeling [64], [65]. The aim of the task is to label a single designated cluster in the MD data by brushing them on the 2D MDP represented by a monochrome scatterplot. We pick the task as it is widely used to explore MD data in visual analytics [64], [65], [66], and is an important use case of previous brushing techniques [6], [26].

Procedure. One experimenter manages the experiment for all participants individually and in person. After a participant signs the consent form, the experimenter explains the concept of MD data and why 2D projections cannot precisely depict them. Then, the experimenter details the tasks and goals of the experiment. During the introduction, the participants are free to ask questions.

Participants are then exposed to 36 trials; each is associated with a single combination of the independent variables. We divide the trials into four sessions, where each session is assigned to a single Technique. In each session, after the experimenter demonstrates the technique, participants are given a maximum of five minutes to practice and pose questions. For the practice, we use a dataset and projections different from those used for the main study (detailed below). The order of the sessions (i.e., the order of the Techniques) is counterbalanced using a four-level Latin square design (Appendix F). Each of these sessions contains nine trials (every combination of DistortionAmount and GlobalInspection), where the order of these trials is again counterbalanced using a Latin rectangular design (Appendix F). As we notice that each trial takes at most around 120 seconds in a pilot study, we set no specific time limit for each trial.

After the sessions, we conducted a semi-structured post-study interview to ask about the perceived usability of brushing techniques and task difficulty (listed in Appendix A). The experiment took less than 50 minutes for all participants.

Measurements. We record labeling results and task completion time of each trial. We record the task completion time as the duration from the first mouse hover to the stimuli to the click of the "finish trial and proceed" button. We also convert labeling results into accuracy scores with respect to the ground-truth clusters and save them. We first compute the precision and recall in terms of extracting the data points belonging to the designated cluster and use the F1 score as a final accuracy.

MD dataset and ground truth clusters. We use the MNIST dataset as a reference MD dataset to generate projections that will be used as scatterplot stimuli. We reduce the dimensionality of the MNIST dataset to 10D using PCA [67] to prevent previous brushing techniques that rely on convex-shaped MD region (Data-driven brushing, *M*-ball brushing) to suffer from the curse of dimensionality [68] (see Appendix B), thereby making our experiment fair.

We consider each class (i.e., digit) as a potential ground truth cluster. We thus render points as snippet images (i.e., digits) so that participants can visually distinguish different digits and readily determine the boundary of brushing.

However, classes may not be well separated in the original MD space [24], [45]. Therefore, we first identify class pairs with high separability following Aupetit [45] to create valid ground-truth clusters and allow reliable evaluation. We first compute the separability of class pairs over 96 labeled MD datasets available using the between-dataset Calinski-Harabasz (CH_{btwn}) Index [44], then pick the index value corresponding to the 90th percentile as a separability threshold. We use CH_{btwn} as the index is proven to work robustly and fairly regardless of the dimensionality [24], [44] Then, for each stimulus, we select pairs of classes in the MNIST dataset that are mutually separable with a CH_{btwn} separability index higher than the separability threshold.

Confounding variables. We identify and control three confounding variables: the number of points of each cluster, the number of clusters, and the non-triviality of the shape of the cluster designated to be brushed. To control the non-triviality of a cluster, we first fit the Gaussian Mixture model [69] with a single Gaussian distribution to the cluster in the MD space. We then use the maximum log likelihood score, quantifying how well the Gaussian fits the data distribution as a proxy for non-triviality. We identify the top three classes with high non-triviality and the bottom three classes with low non-triviality of the MNIST dataset as *High* and *Low* non-triviality clusters, respectively, designating the remaining four classes as *Intermediate* clusters. We also prepare three different settings for the number of points (*100*, *150*, and *200*) and clusters (*two, three,* and *four*). The total number of combinations of the confounding variables is $3 \times 3 \times 3 = 27$, but we reduce it to nine using a three-level Latin square design. Note that we maximally equalized the assignment of these nine combinations to all participants (12) and trials (36) using Latin rectangle design (See Appendix F for the detailed assignment of confounding variables), so that confounding variables can make minimal impact on the study results.

Stimuli (i.e., MDPs) generation. We have nine combinations of confounding factors and three different DistortionAmount settings; we thus need $9 \times 3 = 27$ types of stimuli. When a trial requires a stimulus with $\mathscr{N}$ clusters, $\mathscr{M}$ points per cluster, $\mathscr{L}$ non-triviality, and $\mathscr{O}$ DistortionAmount, we first randomly sample $\mathscr{N}$ clusters while ensuring that at least one of them has $\mathscr{L}$ non-triviality. We then randomly sample data points to make each cluster have $\mathscr{M}$ points and generate a projection having $\mathscr{O}$ DistortionAmount (detailed below) as a final stimulus. One of the clusters with $\mathscr{L}$ non-triviality is randomly designated to be brushed. As we use the MNIST dataset, we designate the cluster by informing the corresponding class. All stimuli are precomputed before the experiment. Please refer to Fig. 5 for the example projections we use.

Levels of DistortionAmount. We generate a projection with *Low distortion* using *t*-SNE [16]. We optimize two hyperparameters that significantly impact the projection results, perplexity, and learning rate [70], to generate projections with the least distortion. We use Bayesian optimization [71] while using the Trustworthiness & Continuity (T&C) [33] as the target function. Specifically, we compute both Trustworthiness and Continuity individually and compute their F1 score. We use T&C as they are widely used metrics for detecting FN and MN [36], [24], [14], [72], respectively. We also create a projection with *High FN* using the random linear projection technique. As the random linear projection always decreases the distances between points, we can expect the resulting projection to have relatively high FN but relatively low MN. Finally, we create a projection labeled *High MN* by executing *t*-SNE with an extremely low perplexity setting of 1. Given that low perplexity values cause *t*-SNE to prioritize local structure, setting the perplexity this low can result in the algorithm missing close neighbors and splitting cluster apart, thereby leading to more MN [73]. We verify the validity of our settings in Appendix E by reporting the T&C scores of the resulting projections.

Training session for each brushing technique. We use PCA projection of a randomly driven set of three highly separated classes for the training. We use PCA as it generates FN distortions but less than random projection. Therefore, using PCA, participants can experience each brushing technique's capability to address distortions without being frustrated by the task's difficulty. We turn on both global and local inspection during the training session. We inform participants that these functionalities can be turned off during the main experiment.

Participants. We recruit 12 participants (identified as E1P1–E1P12) from a local university (nine males and three females, aged 22–32 [26.4 ± 3.2]). All participants had undergraduate-level experience in statistics or linear algebra and thus could readily understand the concept of MD data and brushing techniques. They also had experience in using standard brushing techniques like lasso or box selections. We compensated each participant with the equivalent of US $10 for their participation.

Apparatus. We execute all brushing techniques in a Chrome web browser running on an Apple Macbook Pro 2021 (Apple M1 Max, 64GB RAM). Participants use techniques through a 27-inch 4K monitor with a regular mouse and keyboard setting. We locate the stimuli in a 1000px $\times$ 1000px box at the center of the screen.

5.1.2 Quantitative Results

We detail our study findings, which are also depicted in Fig. 6.

Analysis on overall task accuracy. Aligned with the main study objective (Sect. 5.1.1), we investigate how three independent variables (DistortionAmount, DistortionInspection, and Techniques) affect clustering accuracy, i.e., F1 score (Fig. 6 left). For that purpose, we execute the three-way repeated measure ANOVA (RM ANOVA) [74]. We use Bonferroni correction for post-hoc analysis.

As a result, we find significant main effect on Techniques ($F_{3,33} = 36.361$, $p < .001$) and DistortionAmount ($F_{2,22} = 17.102$, $p < .001$). For Techniques, post-hoc analysis reveals

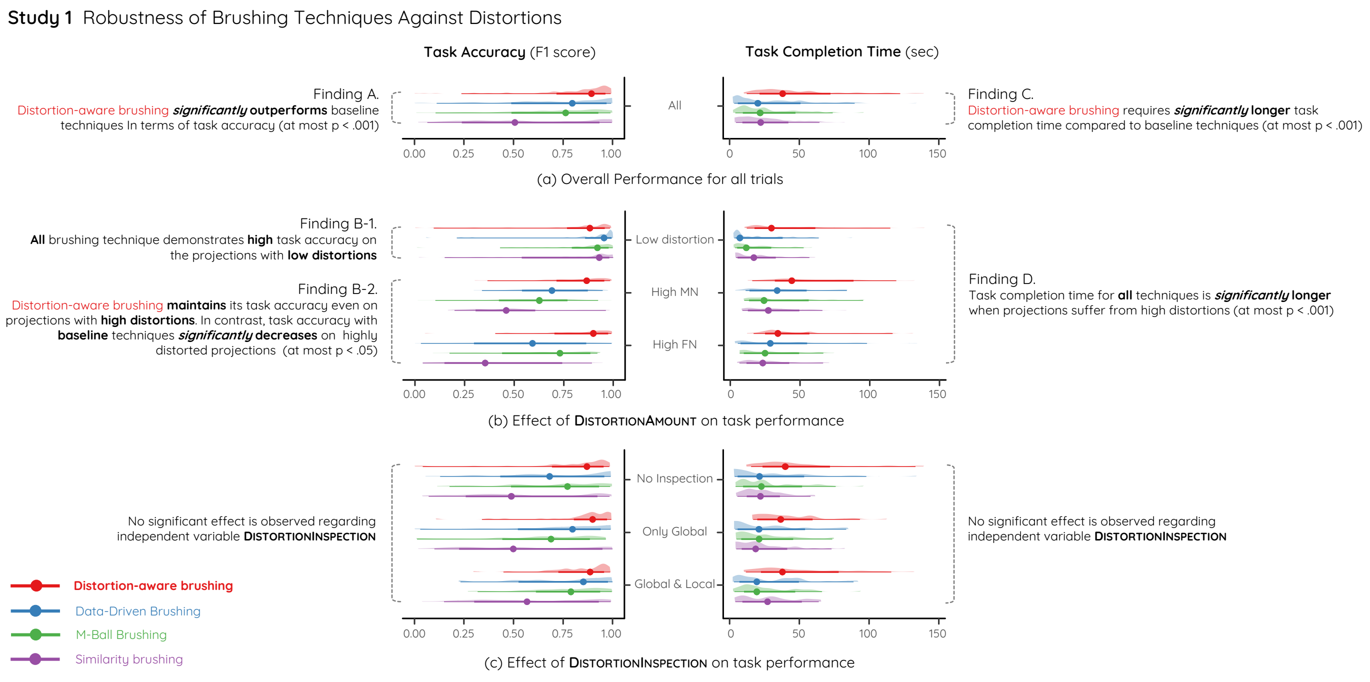

Fig. 6. Study 1 results demonstrating the robustness of brushing techniques against the amount of distortions. Overall, Distortion-aware brushing significantly outperforms baseline techniques in terms of task completion time, but requires participants more time to complete the task. Please refer to Sect. 5.1.2 and Sect. 5.1.3 for detailed analysis results and discussions, respectively.

that Distortion-aware brushing shows significantly higher accuracy compared to baseline techniques ($p < .01$ for Data-driven brushing case, $p < .001$ for other cases). There was no other significant difference. This leads to our first finding:

Finding A. *Distortion-aware brushing is, overall, the most accurate technique among all competitors for selecting clusters in MDPs.*

In terms of DISTORTIONAMOUNT, we find that task accuracy is significantly better on the projections with low distortions compared to the ones with high MN or high FN ($p < .001$ for both cases), which is straightforward.

Interaction effect analysis on task accuracy. Using RM ANOVA, we also find a significant interaction effect between TECHNIQUES and DISTORTIONAMOUNT ($F_{6,66} = 7.272$, $p < .001$). To deep dive into this interaction, we divide the trials into three groups with different DISTORTIONAMOUNT (*Low distortion*, *High MN*, and *High FN*). We then conduct additional one-way ANOVA examining how TECHNIQUES affect the F1 accuracy scores of each group.

As a result, we find that TECHNIQUES variable significantly influences the accuracy with the projections having *High MN* ($F_{3,140} = 20.764$, $p < .001$) or *High FN* ($F_{3,140} = 19.318$, $p < .001$). However, it has no significant effect for the case with *Low distortions* ($F_{3,140} = 0.847$, $p = .470$), where all techniques show substantially high performance (over 0.85 in average; Fig. 6b left). Post-hoc analysis shows that if projections have *High MN* or *High FN*, Distortion-aware brushing significantly outperforms baseline techniques (*High MN*: $p < .01$ for Data-driven brushing case, $p < .001$ for other cases; *High FN*: $p < .05$ for M-Ball brushing case, $p < .001$ for other cases). Fig. 6b (right) shows that such change occurs as the accuracy of Distortion-aware brushing stays still while the accuracy of other techniques decreases. The results can be summarized as follows:

Finding B-1. *Every brushing technique we considered is accurate for the MDPs with low distortions.*

Finding B-2. *Distortion-aware brushing maintains its high task accuracy on projections with high MN and FN. On the other hand, baseline techniques work poorly on these projections, having significantly lower task accuracy compared to Distortion-aware brushing.*

No other significant interaction effect has been found.

Analysis on overall task completion time. We investigate how three independent variables affect task completion time (Fig. 6 right). As with previous analysis on task accuracy, we run the three-way RM ANOVA for that purpose, and use Bonferroni correction for post-hoc analysis.

The analysis reveals significant main effect on TECHNIQUES ($F_{3,33} = 8.557$, $p < .001$) and DISTORTIONAMOUNT ($F_{2,22} = 7.162$, $p < .01$). For TECHNIQUES, post-hoc analysis shows that Distortion-aware brushing shows significantly longer task completion time compared to baseline techniques ($p < .001$ for all cases). This summarizes into:

Finding C. *Distortion-aware brushing requires significantly longer task completion time than baseline techniques.*

In terms of DISTORTIONAMOUNT, post-hoc analysis reveals that the techniques require longer completion time for the projections with *High MN* and *High FN* compared to the *Low distortions* cases. As no interaction effect coupled with DISTORTIONAMOUNT exists, the results lead us to the following finding:

Finding D. *Regardless of the type of brushing techniques, the task completion time was significantly longer when there exist distortions in MDPs.*

Interaction effect analysis on task completion time. RM ANOVA indicates that there exists significant interaction effect on *DistortionAmount* and *DistortionInspection* ($F_{4,44} = 4.236$,$p < .01$). For follow-up analysis, we divide the trials into three groups based on *DistortionAmount* and run one-way ANOVA on each group, investigating the influence of *DistortionInspection* on task completion time. For all groups, we find no significant main effect found (*Low distortion*: $F_{2,141} = 1.368$, $p = 0.258$; *High MN*: $F_{2,141} = 0.340$, $p = 0.712$; *High FN*: $F_{2,141} = 0.779$, $p = 0.461$).

Visual analysis of baseline techniques' task accuracy. We visually explored additional factors affecting the interaction between TECHNIQUES and DISTORTIONAMOUNT (Fig. 6b). Although the difference is not statistically significant, we observe that the median task accuracy of *M*-Ball brushing (green) is higher than the one of Data-driven brushing (blue) for the projections with *High FN*, while the opposite happens for the ones with *High MN*.

5.1.3 Discussions

We discuss the takeaways from our main findings (Sect. 5.1.2).

Distortion-aware brushing is more accurate than competitors when more distortions exist. Findings A and B verify that Distortion-aware brushing significantly outperforms baseline techniques, showing substantial accuracy regardless of distortions. In contrast, baseline techniques work poorly when the projections suffer from distortions. These findings thus clearly imply the benefit of using Distortion-aware brushing in MD data analysis. Distortion-aware brushing will help analysts to reliably analyze the cluster structure of MD data even if MDPs have unreliable representations due to unavoidable distortions (Sect. 1, 2.1).

Baseline techniques can still be used in low-distortion MDP. Our findings also indicate that baseline techniques can still be useful in some situations. Finding B-1 informs that baseline techniques are reliable if we can ensure that the projections have low distortions. In such a case, the analysis will benefit from the faster completion time of baseline techniques (Finding C). However, this will require analysts to detect areas with sufficiently low distortions to be trustworthy [19], or to invest more computational efforts to improve MDPs accuracy (e.g., by optimizing hyperparameters).

Local and global inspections are helpful when distortions are high. The study does not reveal a significant influence on the existence of our inspection functionalities (Step 1 Step 2). However, in the interaction effect analysis on task completion time, we can observe that F-value of ANOVA decreases for *High MN* and *High FN* case, which means that the influence of global and local inspection increases when MDPs suffer from more distortions. Although not supported by statistical evidence, this observation aligns with the qualitative feedback from participants (Sect. 5.3.2) that our global and local inspection functionalities help users perform brushing tasks faster.

Trade-off between task accuracy and task completion time. Although Distortion-aware brushing shows the best task accuracy, it trades task completion time for such achievement (Finding C, D). Our qualitative interview reveals that this tradeoff mainly originates from the enhanced concentration or cautiousness of participants while using Distortion-aware brushing. We detail this phenomenon in Sect. 5.3.1.

Validity of our study design. The higher task accuracy of M-Ball brushing compared to Data-driven brushing in the *High FN* case and the lower accuracy in the *High MN* case reaffirms the validity of our study design and results. Indeed, these findings align with the techniques' design. M-Ball brushing avoids FN by selecting only true neighbors among points within a 2D circular region, but the corresponding MD ball is not as extended as the MD box of the Data-driven brushing in the 10-dimension data space [75], more likely failing to capture some true neighbors missing in the locally brushed layout. In contrast, Data-driven brushing captures more of the missing neighbors but also most of the FN enclosed in its 2D box. Please refer to Appendix B for the detailed design of these techniques.

5.2 User Study 2: Robustness Against the Non-Triviality of Multidimensional Cluster Shape

5.2.1 Objectives and Design

We aim to investigate the robustness of Distortion-aware brushing in maintaining user performances (task accuracy and completion time) against the non-triviality of cluster shapes, comparing it with previous techniques. We also want to check whether the general findings about the performance of Distortion-aware brushing we find in Study 1 (Sect. 5.1) are maintained. To this end, the study has:

- Non-triviality of the MD shape of a designated cluster to be brushed (NONTRIVIALITY)
  - *High*, *Intermediate*, and *Low*,

as an independent variable instead of DISTORTIONAMOUNT. The amount of distortions is controlled for as a confounding variable, along with the number of points and clusters. The study also shares with the previous one the independent variables GLOBALINSPECTION and TECHNIQUES. All other experimental settings are the same as for Study 1. In summary, this study is identical to the previous experiment except that we swapped DISTORTIONAMOUNT for NONTRIVIALITY, and we work with different sets of participants.

Participants. We recruit 12 participants (identified as E2P1–E2P12) from two local universities (ten males and two females, aged 21–30 [24.6 ± 2.5]), again ensuring that all participants have undergraduate-level experience in statistics or linear algebra.

5.2.2 Quantitative Results

The following are the findings from the study (Fig. 7). We first check whether our findings from the previous study which are not dependent on DISTORTIONAMOUNT are maintained in this study (Finding A and C). We also discuss new findings from the study.

Analysis on overall task accuracy. By executing three-way RM ANOVA examining the influence of three independent variables on the F1 score, we find a significant main effect on TECHNIQUES ($F_{3,33} = 34.51$, $p < .001$). Successive Bonferroni post-hoc analysis reveals that Distortion-aware brushing significantly outperforms all other techniques ($p < .001$ for all) in terms of accuracy, again confirming Finding A.

Here, unlike Study 1, post-hoc analysis indicates that Data-driven brushing and M-Ball brushing show significantly higher accuracy compared to Similarity brushing ($p < .001$ for both). This leads to a new finding:

> ***Finding E.*** *Using similarity brushing leads to lower accuracy compared to not only Distortion-aware brushing but also Data-driven brushing and M-Ball brushing.*

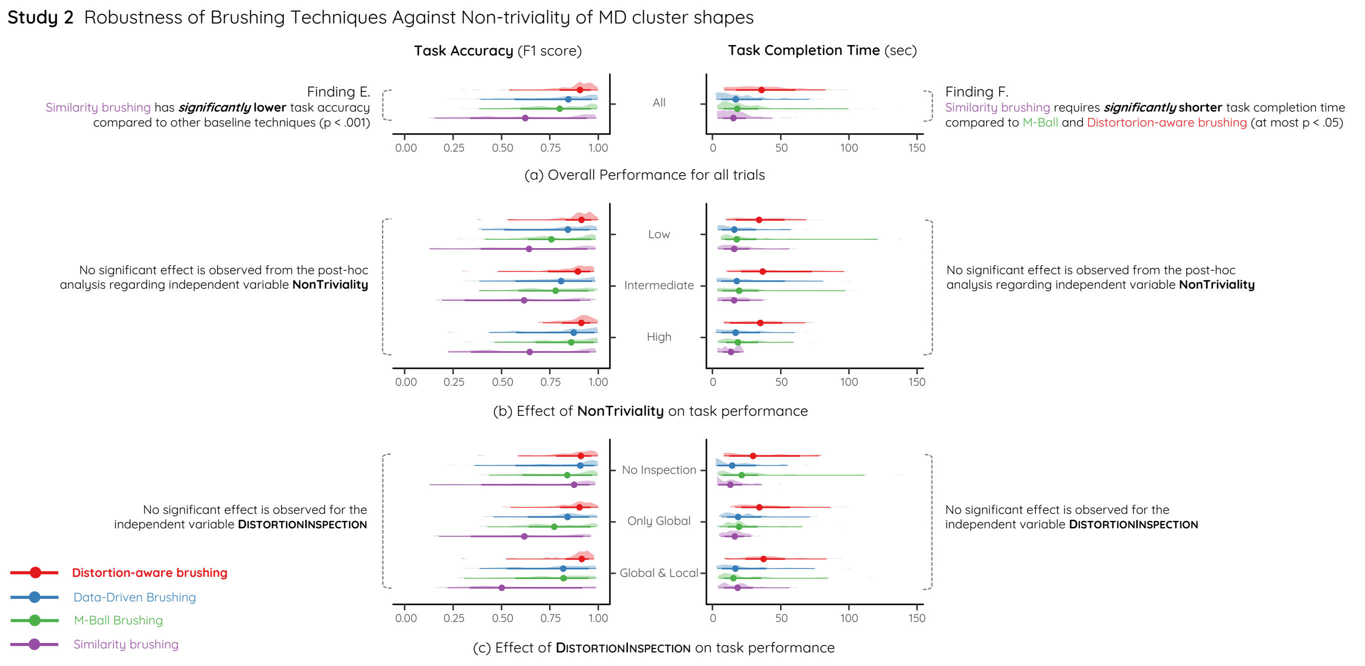

Fig. 7. Study 2 results demonstrating the robustness of brushing techniques against the non-triviality of MD cluster shapes (NONTRIVIALITY). The study results reaffirm the superiority of Distortion-aware brushing in terms of task accuracy. We find no significant differences between brushing techniques in terms of NONTRIVIALITY.

We find a significant main effect for NONTRIVIALITY (($F_{2,22} = 8.55$, $p < .01$), but post-hoc analysis fails to identify significant differences between brushing techniques.

Analysis on overall task completion time. We run three-way RM ANOVA to investigate how three independent variables affect task completion time (Fig. 7 right), which is followed by Bonferroni correction for post-hoc analysis.

RM ANOVA reveals a significant main effect on TECHNIQUES variable (($F_{3,33} = 13.84$, $p < .001$)). The post-hoc analysis indicates that Distortion-aware brushing requires a significantly longer task completion time compared to all other techniques ($p < .001$ for all). This result again confirms Finding C from the previous study. Moreover, the post-hoc analysis finds that the task completion time with M-Ball brushing is significantly longer than Similarity brushing ($p < .05$), hence the finding:

Finding F. *Using similarity brushing leads to shorter task completion time compared to not only Distortion-aware brushing but also M-Ball brushing.*

We again find a significant main effect for NONTRIVIALITY (($F_{2,22} = 5.22$, $p < .05$), but post-hoc analysis identifies no significant differences between brushing techniques.

5.2.3 Discussions

Study 2 again confirms the superiority of Distortion-aware brushing regarding task accuracy and the tradeoff between task accuracy and completion time. Moreover, it also reaffirms that there are no significant effects relevant to DISTORTIONINSPECTION (See Sect. 5.1.3 for detail). The following are the takeaways from the new findings of Study 2 (Sect. 5.2.2).

Superiority and Inferiority between baseline techniques. Finding E shows that M-Ball Brushing and Data-Driven Brushing have better task accuracy than Similarity brushing. Finding F then shows that M-Ball brushing requires longer task completion time compared to Similarity brushing.

Again, these results are consistent with the techniques' design. M-Ball brushing allows the selection of true neighbors of a point, then of true neighbors of these selected points, progressively and manually expanding the brush while avoiding FN distortions, trading time for accuracy. In contrast, Similarity brushing selects all at once the true neighbors of all the points in the painter, which are not necessarily true neighbors of each other (FN), hence trading accuracy for time. Data-driven brushing also captures FN but does not automatically extend the brush to its true neighbors, resulting in fewer distortions than Similarity brushing. Our qualitative analysis of interview results (Sect. 5.3) further verifies the finding that Data-driven brushing showed better task accuracy compared to Similarity brushing.

5.3 Post-Study Interview

We discuss the post-study interview results and takeaways. First, two authors independently perform semantic coding and merge their codebooks. While conducting coding, the authors especially focus on revealing not only the pros of Distortion-aware brushing but also their cons. The final codebook is then formulated by three iterations of discussions. As a result, we identify three themes: (1) benefits of Distortion-aware brushing in user experience, (2) benefits of global and local inspection functionalities, and (3) possible improvements for Distortion-aware brushing.

5.3.1 Benefits of Distortion-aware brushing in Terms of User Experience

Our interview reveals Distortion-aware brushing's positive effects on user confidence, cautiousness, and serendipity.

Distortion-aware brushing makes users more confident. Participants report that Distortion-aware brushing makes them more confident. Specifically, 20 out of 24 participants (83%) report that they are most confident when using Distortion-aware brushing.

We find that such benefit mainly originates from the fact that Distortion-aware brushing allows users to make final decisions about adding new points to the brush (Step 4). In contrast, baseline techniques automatically formulate the MD region and only "notify" users of the points within the region being brushed points. 10 out of 24 participants (42%) explicitly claim they are not able to manually fine-tune the brushed points in the baseline techniques, which leads them to lower confidence.

We also find that participants feel more confident when using Distortion-aware brushing as they are able to see an MD cluster as a 2D cluster (O1). Eight out of 24 (33%) participants mention that they are more confident about their interactions as the visual 2D cluster informs that they are doing well. For example, E1P12 note, *"[While using Distortion-aware brushing], I felt quite confident and even relieved when I saw that there were only a few other data points [which do not represent a designated digit] included in my cluster"*. On the other hand, participants are not able to visually confirm the correctness of their interaction in baseline techniques, which makes them less confident.

Distortion-aware brushing makes users more cautious. Our interview reveals that the enhanced confidence again derived from the enhanced cautiousness of participants. E2P5 noted, *''As I knew that I was doing well, I wanted to perform my best in performing the given task"*. The result also aligns well with our quantitative findings. It is intuitive that people will require more time to complete tasks when they are more cautious (Finding C), but this will result in better accuracy (Findings A, B). Still, being more cautious can make users more tired; balancing between cautiousness and ease of use will be an interesting future work.

Serendipity in using Distortion-aware brushing. An unexpected benefit of Distortion-aware brushing is that users enjoy and have fun using it. We find that 11 out of 24 participants (45%) note that playing with Distortion-aware brushing is fun, and five among them (21%) especially use the word "game" to explain their serendipity. E1P6 noted, *"I feel like playing a game that captures small monsters. I want to achieve a higher score."*. Enhancing Distortion-aware brushing by strengthening its gamification feature will be an interesting future work.

5.3.2 Benefits of Global and Local Inspections

Although not statistically validated, our studies provide evidence that global and local inspection functionalities (Step 1 Step 2) provide a positive effect in reducing task completion time (Sect. 5.1.3). Aligned with this result, our interview also provides qualitative evidence of the positiveness of local inspection functionality. Seven out of 24 participants (29%) reported that the local inspection functionality substantially helps them quickly determine which action to execute next. However, no participants explicitly indicate the benefit of global inspection functionality. We believe this is primarily because image snippets already support easy identification of the correct starting point for brushing. Examining the effect of global inspection without image snippets will be an interesting future avenue to explore.

5.3.3 Possible Improvements of Distortion-aware brushing

The interview also reveals limitations and possible improvements for Distortion-aware brushing.

Steepen the learning curve. Participants report that it is not cognitively difficult to understand the workflow of Distortion-aware brushing, but it still takes some time to learn the technique. While only three out of 24 participants (13%) mentioned that the complexity of Distortion-aware brushing impeded their task, 16 out of 24 participants (67%) mentioned that some practice is needed to fully understand and utilize the functionalities of Distortion-aware brushing. Making Distortion-aware brushing easier to learn will be an important future work that potentially enhances the applicability of the technique.

Enhance Familiarity. In the interview, ten out of 24 participants (42%) mentioned that Data-driven brushing is the most comfortable technique as they are familiar with brushing data points using box-shaped regions. They indicate being less accustomed to painter-based brushing (which is the case for Distortion-aware brushing and Similarity brushing), making it more difficult to accurately control the painter. The fact that Distortion-aware brushing outperforms all other baseline techniques in terms of task accuracy shows the superiority of our point relocation scheme. Nevertheless, leveraging box-shaped region to brush points in Distortion-aware brushing would likely further improve the technique both in terms of task accuracy and completion time.

6 Use Case

We demonstrate a use case validating Distortion-aware brushing's effectiveness in solving real-world analytic problems.

6.1 Persona and Goals

We define a persona named Alice, a data analyst hired by a casual dining franchise company. The company wants to open new restaurants in California, but investigating the profitability of every block in the state exceeds its budget. Thus, the company asked Alice to find good candidate areas for detailed examination with three constraints: **(C1)** report blocks that are geographically similar to each other, which will substantially reduce the investigation cost. **(C2)** report blocks that are similar to each other for many attributes (i.e., well clustered in the MD space), also for reducing the investigation cost. **(C3)** focus on the blocks with higher average home prices, as people who have the financial means to pay for the restaurant's food may live in such areas.

6.2 Equipment

Dataset. Alice used California Housing dataset [76], comprised of nine attributes (*Longitude*, *Latitude*, *Median Income*, *House Age*, *Average Rooms*, *Average Bedrooms*, *Population*, *Average Occupancy*, and *House Price*). Each datum corresponds to an individual block, which is the smallest geographical unit used in the U.S. census. Each attribute value is a statistic that summarizes all households in a corresponding block.

System design. Alice used a visual analytics system leveraging Distortion-aware brushing consisting of three components: Brushing View, Parallel Coordinate (PC) View, and Attribute View.

Brushing View provides Distortion-aware brushing, where 2D projection is made by mapping longitude and latitude to x and y axes, and the map of California is displayed in the projection background. The MD space is set as the 7D space formed by all other attributes. Note that the background map is hidden when the points are relocated (Steps 3, 4), as 2D positions of data points have no more direct relation to the map. When users go back to

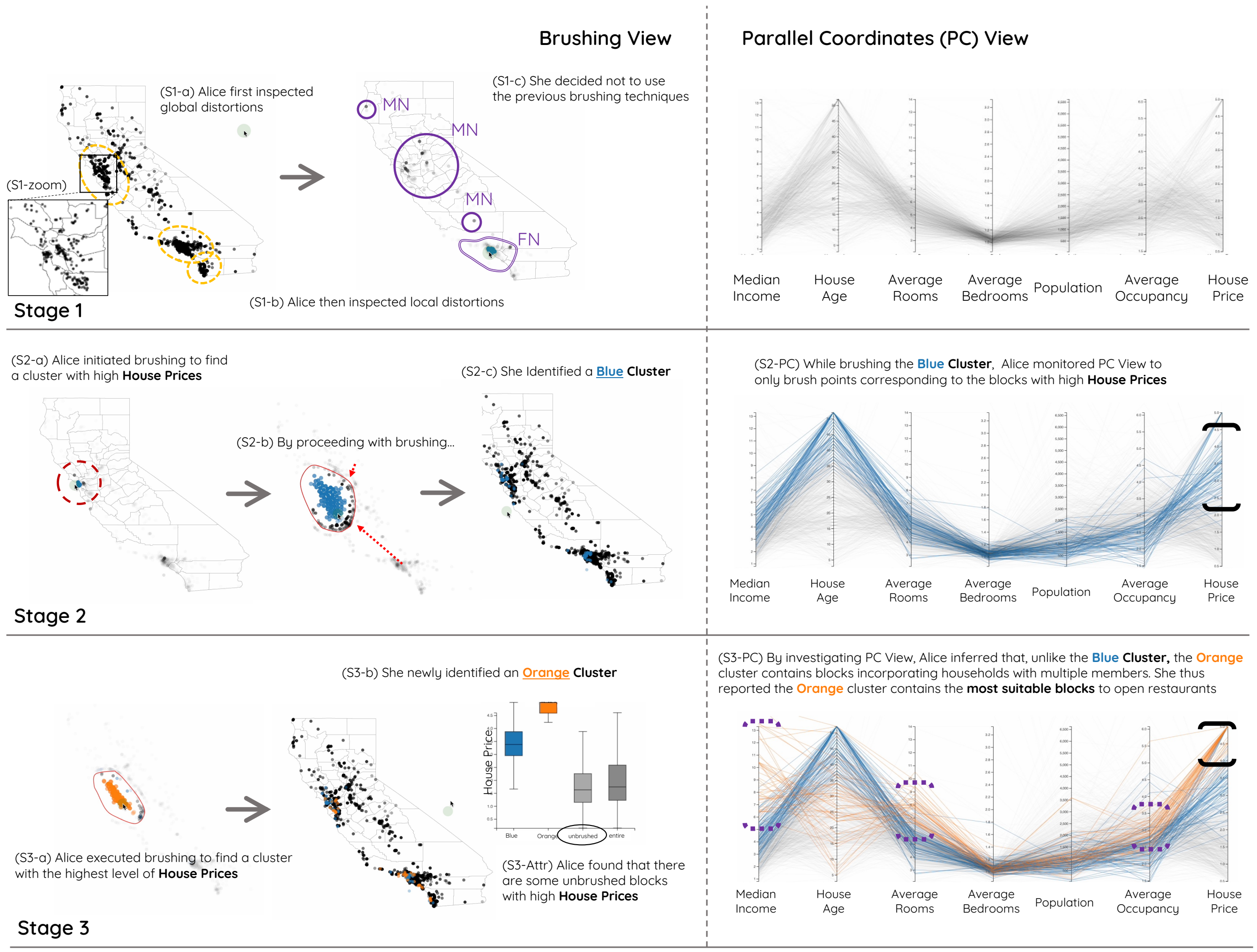

Fig. 8. Use case of Distortion-aware brushing, using the technique to explore the California Housing Dataset [76] (Sect. 6) to find a good candidate region for opening casual dining restaurants. Our persona, Alice, used a visual analytics system composed of a brushing view (left) that serves Distortion-aware brushing and a Parallel Coordinates (PC) view (right) that shows the attribute values of brushed points. Alice also checked detailed statistics of attribute values in brushed points by checking the Attribute (Attr) view. Using a visual analytics tool incorporating Distortion-aware brushing, our persona, Alice, extracted clusters of blocks with similar characteristics and successfully conducted the given request.

the original projection using contextualization (Sect. 4.2.3), they can see the map again.

PC View helps users to directly monitor how the brushing proceeds and get hints about initiating or terminating brushing. PC dynamically reacts to user interaction in the Brushing View; the lines corresponding to seed points while inspecting local distortions (Step 2), and the brushed points (Step 4) are highlighted using the same color as the points. All other points are depicted in black with lower opacity.

Finally, the Attribute (Attr) View displays the distribution of attribute values of (1) brushed clusters, (2) remaining points, and (3) all points using boxplots. Boxplots are dynamically updated, reflecting the brushing status in the Brushing View.

6.3 Scenario

(Stage 1) Inspecting local distortions

Alice first wanted to examine whether she could report geographically similar points as clusters (C1). To do so, she checked whether visual proximity between 2D points matches the MD similarity. By inspecting the global distortion using density encoding (Step 1), she found several 2D clusters in the projection (i.e., map) that have high MD density (yellow dotted ellipses in Fig. 8 S1-a). Note that she zoomed in to remove clutter and verified there are points with high MD density in the places (Fig. 8 S1-zoom). Thus, she hypothesized that these regions may contain blocks that can also be considered clusters in the MD space.

To validate her hypothesis, Alice hovered the painter around three regions to examine their local distortion. As a result, she found that all regions suffer from both FN (neighboring map locations having low MD similarity, depicted by low opacity; purple doubled line in Fig. 8 Stage 1) and MN (map locations far apart having similar attributes; purple solid line in Fig. 8 Stage 1), rejecting the hypothesis (Fig. 8 S1-b). This means that Alice *cannot use conventional brushing or previous brushing techniques for MDP* (Sect. 2) as they define the 2D brush as a compact, continuous 2D region vulnerable to FN. Therefore, Alice reported that it is difficult to satisfy C1 and decided to keep using Distortion-aware brushing for the following analysis (Fig. 8 S1-c).

(Stage 2) Extracting a cluster with high house prices

Alice then aimed to find points having high *House Prices* (C3) and being well clustered (C2). She first moved the painter around the projection while monitoring seed points' attribute values using

PC view. As she expected that such "wealthy regions" may be located in urban areas, she especially examined regions around Los Angeles and San Francisco. As a result, she found an area that contains seed points having *House Prices* ranging from 250,000$ to 400,000$, which exceeds the median house price, near San Fransisco downtown (red dashed circle in Fig. 8 Stage 2; S2-a). She started brushing from the seed points while monitoring the PC View (Steps 3, 4) (Fig. 8 S2-b), and terminated brushing when the brush started to contain the blocks that have significantly different patterns in PC View. As a result, she brushed the blocks with *House Prices* lower than approximately 200,000$ (black solid bracket in Fig. 8 S2-PC). Following the corresponding brush color, she named the cluster as a *Blue cluster* (Fig. 8 S2-c). Note that this *cannot be done by filtering high-price blocks in PC View* as she does not know the threshold of *House Prices* that can accurately discriminate blocks having different patterns in PC View.

(Stage 3) Extracting a cluster with the highest house prices

Through the PC View and the Attribute View, Alice discovered that the portion of blocks with the highest *House Prices* are not incorporated in the Blue cluster. Therefore, she decided to find data points with similar attribute values (C2) that may comprise these highly-priced blocks (C3). By skimming through the urban area again, she found seed points with the highest level of *House Prices* and started brushing again (Fig. 8 S3-a). To clearly discriminate the currently brushed cluster from the Blue cluster, she stopped brushing when the newly brushed cluster started to incorporate the blocks with prices lower than 400,000$ (black solid bracket in Fig. 8 S3-PC). She named the cluster the *Orange cluster*, following the same naming convention (Fig. 8 S3-b).

Alice found that the orange cluster not only exceeds the blue cluster in terms of *House Prices* but also in *Median Income*, *Average Room*, and *Average Occupancy* (purple dotted brackets in Fig. 8 S3-PC). Higher *Average Occupancy* denotes that the blocks in the orange cluster may contain households consisting of multiple people. Higher *Median Income* and *Average Room* also support this inference, as they will grow proportionally to the number of members in each household. Alice thus concluded that the *Orange cluster* contains blocks that are more appropriate for opening casual dining restaurants, as such restaurants may not target people who eat alone [77].

Alice thus reported the blocks within the orange cluster as having top priority for investigation. It is worth noting that blue and orange clusters have no clear segmentation in terms of individual attribute values (Fig. 8 S3-PC). This means that the *neither brushing axes of the PC nor previous brushing techniques for MDP cannot precisely reveal these clusters* (see Appendix G for further confirmation). Moreover, points within two clusters are all scattered throughout the projection (Fig. 8 Stage 4). This reaffirms that *both a naive 2D brushing and previous brushing techniques for MDP cannot stand out to detect these clusters*, although these clusters have clear differences overall (S3-PC)—yet a clear benefit of Distortion-aware brushing in this scenario.

7 Discussions

We discuss the usability of Distortion-aware brushing in three aspects: comparison with automatic clustering, use cases beyond MDPs, and scalability. We also discuss the limitations of the study.

7.1 Comparison to Automatic Clustering Techniques

Distortion-aware brushing and automatic clustering techniques (e.g., K-Means) both aim to discover meaningful cluster patterns in MD data. Actually, the design of Distortion-aware brushing's is inspired by density-peak clustering algorithms [58], involving initial identification of high-density seed points (Step 1 Step 2) and the subsequent addition of nearby points (Step 4).

However, by involving users in extracting clusters, Distortion-aware brushing makes cluster analysis more insightful. First, by allowing users to "see" image snippets (Fig. 1) or auxiliary visualizations (Sect. 6), Distortion-aware brushing can regard visual similarity between data items that may not be reflected in clustering techniques that rely on conventional similarity or distance functions (e.g., Euclidean, kNN). Moreover, Distortion-aware brushing *empowers* users by giving them enhanced *control* of the clustering process through *spatially relevant* display of *enriched MD information* while being easy and fun to use. As seen in our use case (Sect. 6), users can incorporate their external knowledge into the clustering procedure, specify characteristics of clusters that they want to discover, and set the boundary of clusters themselves. These functionalities enrich data analysis and are not possible in conventional clustering techniques. Such benefits emphasize both the effectiveness of Distortion-aware brushing and the importance of the interactive aspect in cluster analysis.

7.2 Use Cases Beyond MDPs

This research contributes Distortion-aware brushing to make *cluster analysis of MD data more reliable*. Our studies and usage scenario align with this purpose.

However, the core concept of Distortion-aware brushing can be applied beyond MD data analysis and their projections. Indeed, Distortion-aware brushing can be applied to explore and cluster all kinds of data entities that have discrepancies between their 2D spatial positions and semantic meanings. For example, it can be used to browse computer files within messy directories (e.g., `Downloads` or `Desktop`) by interactively clustering the files based on the similarity of their contents. In this case, users can be informed by not only the snippet image of each file (i.e., icons) but also by invisible semantic distances between file contents. It can also be used to explore and cluster a large corpus of chart images based on their semantics (e.g., chart types, data attributes, etc.). We plan to test these use cases in the future, enlightening the potential of not only Distortion-aware brushing but also other visual analytics techniques in aiding our daily lives.

7.3 Visual and Computational Scalability

Distortion-aware brushing's computational complexity in lens update and point relocation is $O(m \log m + \kappa nm)$, where n and m denote the total number of points and the size of the convex hull, respectively (we detail the complexity analysis in Appendix C), and κ denotes the number of nearest neighbors considered to compute closeness. As $n \gg m$ and $n \gg \kappa$, the running time of the technique is mainly bounded by n, making it highly scalable. Note that the only parameter of the technique is κ, which is set by the user tuning the radius of the painter using the mouse wheel.

However, the technique may suffer from visual complexity. When applied to a large dataset, visual clutter can hide image snippets and opacity encoding, making it hard to identify good points to start brushing or properly confirm brushed results.

Enhancing Distortion-aware brushing's encoding to mitigate such visual complexity will be an interesting future work. For example, we may adopt density plots [78], [79] to resolve the clutter or consider brushing a representative subset of the data, then use automatic classification to group the remaining data, similar to the semi-automatic approach proposed with arrangement and grouping of image data [80], [81].

7.4 Limitations

Handling soft clusters. In this research, we assumed "hard clusters", which means that every point can be assigned to at most one cluster. In reality, soft clusters are often observed; a point can belong to multiple clusters with a certain degree. Distortion-aware brushing considers soft clustering only in a within-cluster manner: the users can decide the degree of closeness of points to be brushed. Enhancing the technique to handle soft clusters will be an interesting future research.

Limitations in point relocation. The core concept of point relocation is to resolve local distortions by building a brush that reflects an MD cluster (O1) at the cost of increasing distortions in other areas. Therefore, users can robustly brush MD clusters regardless of MDP distortions (O2).

However, this approach has a downside: it sacrifices the global context provided by the original MDP. Users may struggle to perceive the connection between the original projection and the brushes, potentially reducing the performance of user tasks on global structure investigation, e.g., comparing the density of clusters [5]. An easy solution is to juxtapose the interactive map with the original map, trading visual space for contextual information.

Still, by compromising on global reliability, users can gain substantially high reliability in extracting and examining local clusters, as demonstrated in our user study (Sect. 5). This advantage enables Distortion-aware brushing to reliably support the typical data analysis flow of the visual seeking mantra [82]: *Overview first, zoom and filter, details on demand*. After users inspect the *overview* of data distribution through the original MDP, they can accurately *zoom into* or *filter* local clusters using the technique, then reliably analyze clusters in *detail* through image snippets or auxiliary visualizations. The strategy to enhance local reliability by sacrificing global reliability is similar to that of Focus-and-Context (FC) (e.g., Fisheye [83]) and magic lens techniques (e.g., Proxilens [52], Moleview [84]). However, while FC approaches only provide transient enhancement of local reliability, Distortion-aware brushing continuously maintains the brushed local MD cluster, which enables more detailed and tangible follow-up analysis (Sect. 6). Another related approach involves only considering interactive topological arrangement and grouping in the layout [80], [81] using interactive Voronoi treemaps (IVT). This approach relies entirely on the snippet image information to generate clusters, ignoring other data feature similarity information. In contrast, Distortion-aware brushing proposes an efficient trade-off: it allows for the free re-arranging of the image layout as IVT does, while also providing the ability to brush over a fixed or MDP-constrained spatial arrangement that inaccurately represents the local data similarities, as seen with the standard brushing techniques.

Limitations in user studies. Our user studies validate the effectiveness of Distortion-aware brushing in accurately capturing MD clusters regardless of distortions and the shape of MD clusters. However, the studies also have several limitations that constrain the generalizability of the results and findings. Data points are visualized using image snippets in our experiment, but data points may lack intuitive visual representations in real-world data analysis, *e.g.* tabular data or text documents. We plan to improve the generalizability of our study by testing the situation in which auxiliary visualizations (e.g., Sect. 6) are used to guide the brushing. We also want to clarify the need to conduct experiments with more datasets. For instance, NONTRIVIALITY may not have a statistically significant effect on the studied brushing techniques because the variation in the non-triviality of cluster shapes between digits in the MNIST dataset was not large enough. Another aspect to consider is that some brushing techniques, like Data-driven brushing, are supposed to be used and logically aggregated over multiple views of the data (*e.g.* SPLOM or PCP), which could make them more robust to FN or MN distortions existing in a single view as in our setting. Our study indicates that if a single view is available, then Distortion-aware brushing is significantly more accurate than the other approaches. The case of multiple-linked views is yet to be explored.

8 CONCLUSION

Although brushing in MDP has long been considered an important research topic, previous brushing techniques for MDPs have struggled to overcome distortions. To tackle this problem, we proposed Distortion-aware brushing, which relocates points to resolve distortions and materialize MD clusters as user-generated visual groupings. Our user studies demonstrated the usefulness and usability of Distortion-aware brushing in exploring and discovering MD clusters, its robustness to distortions, and its greater accuracy compared to state-of-the-art brushing techniques. In summary, our work advances the research community one step closer to making more reliable MD data analysis.

REFERENCES

[1] J. Heer, M. Agrawala, and W. Willett, "Generalized selection via interactive query relaxation," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Syst.*, ser. CHI '08, 2008, p. 959–968.

[2] R. A. Becker and W. S. Cleveland, "Brushing scatterplots," *Technometrics*, vol. 29, no. 2, pp. 127–142, 1987.

[3] R. A. Becker, W. S. Cleveland, and A. R. Wilks, "Dynamic graphics for data analysis," *Statistical Science*, vol. 2, no. 4, pp. 355–383, 1987.

[4] M. A. Fisherkeller, J. H. Friedman, and J. W. Tukey, "Prim-9: An interactive multi-dimensional data display and analysis system," in *ACM Pacific*, 1975.

[5] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, and S. Liu, "Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 28, no. 1, pp. 529–539, 2022.

[6] M. Aupetit, N. Heulot, and J.-D. Fekete, "A multidimensional brush for scatterplot data analytics," in *IEEE Conf. on Visual Analytics Science and Technology (VAST)*, 2014, pp. 221–222.

[7] M. Cavallo and C. Demiralp, "Clustrophile 2: Guided visual clustering analysis," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 25, no. 1, pp. 267–276, 2019.

[8] J. Wenskovitch, I. Crandell, N. Ramakrishnan, L. House, S. Leman, and C. North, "Towards a systematic combination of dimension reduction and clustering in visual analytics," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 24, no. 1, pp. 131–141, 2018.

[9] G. J. Quadri and P. Rosen, "Modeling the influence of visual density on cluster perception in scatterplots using topology," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 27, no. 2, pp. 1829–1839, 2021.

[10] G. J. Quadri, J. A. Nieves, B. M. Wiernik, and P. Rosen, "Automatic scatterplot design optimization for clustering identification," *IEEE Tran. Vis. and Comput. Graphics*, vol. 29, no. 10, pp. 4312–4327, 2023.

[11] H. Jeon, G. J. Quadri, H. Lee, P. Rosen, D. A. Szafir, and J. Seo, "Clams: A cluster ambiguity measure for estimating perceptual variability in visual clustering," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 1, pp. 770–780, 2024.

[12] E. Tejada, R. Minghim, and L. G. Nonato, "On improved projection techniques to support visual exploration of multidimensional data sets," *Inform. Visualization*, vol. 2, no. 4, p. 218–231, Dec. 2003.

[13] S. Bonakala, M. Aupetit, H. Bensmail, and F. El-Mellouhi, "A human-in-the-loop approach for visual clustering of overlapping materials science data," *Digital Discovery*, vol. 3, pp. 502–513, 2024.

[14] L. G. Nonato and M. Aupetit, "Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 25, no. 8, pp. 2650–2673, 2019.

[15] T. Fujiwara, Y.-H. Kuo, A. Ynnerman, and K.-L. Ma, "Feature learning for nonlinear dimensionality reduction toward maximal extraction of hidden patterns," in *2023 IEEE 16th Pacific Visualization Symp. (PacificVis)*, 2023, pp. 122–131.

[16] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. 86, pp. 2579–2605, 2008.

[17] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[18] M. Aupetit, "Visualizing distortions and recovering topology in continuous projection techniques," *Neurocomputing*, vol. 70, no. 7, pp. 1304–1330, 2007.

[19] S. Lespinats and M. Aupetit, "Checkviz: Sanity check and topological clues for linear and non-linear mappings," *Computer Graphics Forum*, vol. 30, no. 1, pp. 113–125, 2011.

[20] S. Lespinats, M. Verleysen, A. Giron, and B. Fertil, "Dd-hds: A method for visualization and exploration of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp. 1265–1279, 2007.

[21] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea, "Toward a quantitative survey of dimension reduction techniques," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 27, no. 3, pp. 2153–2173, 2021.

[22] J. A. Lee and M. Verleysen, "Shift-invariant similarities circumvent distance concentration in stochastic neighbor embedding and variants," *Procedia Computer Science*, vol. 4, pp. 538–547, 2011, proc. of ICCS.

[23] H. Jeon, H.-K. Ko, S. Lee, J. Jo, and J. Seo, "Uniform manifold approximation with two-phase optimization," in *2022 IEEE Visualization and Visual Analytics (VIS)*, 2022, pp. 80–84.

[24] H. Jeon, Y.-H. Kuo, M. Aupetit, K.-L. Ma, and J. Seo, "Classes are not clusters: Improving label-based evaluation of dimensionality reduction," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 1, pp. 781–791, 2024.

[25] R. Motta, R. Minghim, A. de Andrade Lopes, and M. Cristina F. Oliveira, "Graph-based measures to assist user assessment of multidimensional projections," *Neurocomputing*, vol. 150, pp. 583–598, 2015.

[26] M. Novotnỳ and H. Hauser, "Similarity brushing for exploring multidimensional relations," vol. 14, pp. 105–112, 2006.

[27] M. Ward, "Xmdvtool: integrating multiple methods for visualizing multivariate data," in *IEEE Visualization Conf.*, 1994, pp. 326–333.

[28] A. R. Martin and M. O. Ward, "High dimensional brushing for interactive exploration of multivariate data," in *IEEE Vis. Conf.* IEEE, 1995, p. 271.

[29] R. Etemadpour, R. Motta, J. G. d. S. Paiva, R. Minghim, M. C. F. de Oliveira, and L. Linsen, "Perception-based evaluation of projection methods for multidimensional data visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 1, pp. 81–94, 2015.

[30] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner, "Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences," in *the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, 2014, p. 1–8.

[31] R. Etemadpour, L. Linsen, C. Crick, and A. Forbes, "A user-centric taxonomy for multidimensional data projection tasks," in *Proc. of the 6th Int. Conf. on Info. Vis. Theory and Applications*, 2015, pp. 51–62.

[32] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization," *JMLR*, vol. 11, no. 13, pp. 451–490, 2010.

[33] J. Venna and S. Kaski, "Local multidimensional scaling," *Neural Networks*, vol. 19, no. 6, pp. 889–899, 2006.

[34] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. Van Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "Gpgpu linear complexity t-sne optimization," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 26, no. 1, pp. 1172–1181, 2020.

[35] C. Fu, Y. Zhang, D. Cai, and X. Ren, "Atsne: Efficient and robust visualization on gpu through hierarchical optimization," in *ACM KDD*, 2019, p. 176–186.

[36] H. Jeon, H.-K. Ko, J. Jo, Y. Kim, and J. Seo, "Measuring and explaining the inter-cluster reliability of multidimensional projections," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 28, no. 1, pp. 551–561, 2021.

[37] J. A. Lee and M. Verleysen, "Quality assessment of dimensionality reduction: Rank-based criteria," *Neurocomputing*, vol. 72, no. 7, pp. 1431–1443, 2009.

[38] R. M. Martins, D. B. Coimbra, R. Minghim, and A. Telea, "Visual analysis of dimensionality reduction quality for parameterized projections," *Computers & Graphics*, vol. 41, pp. 26–42, 2014.

[39] C. Seifert, V. Sabol, and W. Kienreich, "Stress maps: Analysing local phenomena in dimensionality reduction based visualisations." in *EuroVAST @ EuroVis*. The Eurographics Association, 2010.

[40] N. Heulot, M. Aupetit, and J.-D. Fekete, "Proxiviz: an interactive visualization technique to overcome multidimensional scaling artifacts," *Proc. of IEEE InfoVis (poster)*, vol. 2, 2012.

[41] B. Colange, L. Vuillon, S. Lespinats, and D. Dutykh, "MING: an interpretative support method for visual exploration of multidimensional data," *Inf. Vis.*, vol. 21, no. 3, pp. 246–269, 2022.

[42] L. Meng, S. van den Elzen, N. Pezzotti, and A. Vilanova, "Class-constrained t-sne: Combining data features and class probabilities," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 30, no. 1, pp. 164–174, 2024.

[43] B. Colange, J. Peltonen, M. Aupetit, D. Dutykh, and S. Lespinats, "Steering distortions to preserve classes and neighbors in supervised dimensionality reduction," in *Adv. in Neural Inform. Process. Syst.*, 2020.

[44] H. Jeon, M. Aupetit, D. Shin, A. Cho, S. Park, and J. Seo, "Sanity check for external clustering validation benchmarks using internal validation measures," 2022.

[45] M. Aupetit, "Sanity check for class-coloring-based evaluation of dimension reduction techniques," in *the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, 2014, p. 134–141.

[46] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko, "Dust & magnet: Multivariate information visualization using a magnet metaphor," *Inform. Visualization*, vol. 4, no. 4, pp. 239–256, 2005.

[47] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, "ipca: An interactive system for pca-based visual analytics," *Computer Graphics Forum*, vol. 28, no. 3, pp. 767–774, 2009.

[48] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato, "Local affine multidimensional projection," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2563–2571, 2011.

[49] J. Xia, L. Huang, W. Lin, X. Zhao, J. Wu, Y. Chen, Y. Zhao, and W. Chen, "Interactive visual cluster analysis by contrastive dimensionality reduction," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 29, no. 1, pp. 734–744, 2023.

[50] J. F. Kruiger, A. Hassoumi, H. Schulz, A. C. Telea, and C. Hurter, "Multidimensional data exploration by explicitly controlled animation," *Informatics*, vol. 4, no. 3, p. 26, 2017.

[51] J. Stahnke, M. Dörk, B. Müller, and A. Thom, "Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 22, no. 1, pp. 629–638, 2016.

[52] N. Heulot, M. Aupetit, and J. D. Fekete, "Proxilens: Interactive exploration of high-dimensional data using projections," in *EuroVis Workshop on Visual Analytics using Multidimensional Projections*, Jun 2013.

[53] A. Inselberg, "The plane with parallel coordinates," *The Visual Computer*, vol. 1, no. 2, pp. 69–91, 1985.

[54] R. C. Roberts, R. S. Laramee, G. A. Smith, P. Brookes, and T. D'Cruze, "Smart brushing for parallel coordinates," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 3, pp. 1575–1590, 2019.

[55] H. Hauser, F. Ledermann, and H. Doleisch, "Angular brushing of extended parallel coordinates," in *IEEE Symp. on Inform. Visualization, 2002*, 2002, pp. 127–130.

[56] D. Kammer, M. Keck, T. Gründer, A. Maasch, T. Thom, M. Kleinsteuber, and R. Groh, "Glyphboard: Visual exploration of high-dimensional data combining glyphs with dimensionality reduction," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 26, no. 4, pp. 1661–1671, 2020.

[57] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert, "Axisketcher: Interactive nonlinear axis mapping of visualizations through user drawings," *IEEE Trans. on Vis. and Comput. Graphics*, vol. 23, no. 1, pp. 221–230, 2017.

[58] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[59] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Information Sciences*, vol. 450, pp. 200–226, 2018.

[60] L. Ertöz, M. Steinbach, and V. Kumar, "A new shared nearest neighbor clustering algorithm and its applications," in *Workshop on Clustering*

High dimensional Data and its Applications at 2nd SIAM Int. Conf. on Data mining, 2002, pp. 105–115.

[61] K. Chaudhuri and S. Dasgupta, “Rates of convergence for nearest neighbor classification,” in *Adv. in Neural Inform. Process. Syst.*, vol. 27. Curran Associates, Inc., 2014.

[62] Q. Du, V. Faber, and M. Gunzburger, “Centroidal voronoi tessellations: Applications and algorithms,” *SIAM Review*, vol. 41, no. 4, pp. 637–676.

[63] S. Lloyd, “Least squares quantization in pcm,” *IEEE Trans. on Inform. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[64] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim, “Visual interaction with dimensionality reduction: A structured literature analysis,” *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 241–250, 2017.

[65] J. Peltonen, M. Sandholm, and S. Kaski, “Information retrieval perspective to interactive data visualization,” in *Proc. of the Eurographics Conf. on Visualization (Eurovis 2013)*, 2013, pp. 49–53.

[66] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, and S. Liu, “Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study,” *IEEE Trans. Vis. Comput. Graphics*, pp. 1–1, 2021.

[67] K. F. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[68] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[69] G. Xuan, W. Zhang, and P. Chai, “Em algorithms of gaussian mixture model and hidden markov model,” in *the International Conference on Image Processing*, vol. 1, 2001, pp. 145–148 vol.1.

[70] R. Gove, L. Cadalzo, N. Leiby, J. M. Singer, and A. Zaitzeff, “New guidance for using t-sne: Alternative defaults, hyperparameter selection automation, and comparative evaluation,” *Visual Informatics*, vol. 6, no. 2, pp. 87–97, 2022.

[71] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Adv. in Neural Inform. Process. Syst.*, vol. 25. Curran Associates, Inc., 2012.

[72] H. Jeon, A. Cho, J. Jang, S. Lee, J. Hyun, H.-K. Ko, J. Jo, and J. Seo, “Zadu: A python library for evaluating the reliability of dimensionality reduction embeddings,” in *IEEE Visualization and Visual Analytics*, 2023, pp. 196–200.

[73] M. Wattenberg, F. Viégas, and I. Johnson, “How to use t-sne effectively,” *Distill*, 2016. [Online]. Available: http://distill.pub/2016/misread-tsne

[74] R. Gueorguieva and J. H. Krystal, “Move Over ANOVA: Progress in Analyzing Repeated-Measures Data andIts Reflection in Papers Published in the Archives of General Psychiatry,” *Archives of General Psychiatry*, vol. 61, no. 3, pp. 310–317, 03 2004.

[75] M. Verleysen and D. François, “The curse of dimensionality in data mining and time series prediction,” in *Proc. of ICANN*, 2005, p. 758–770.

[76] R. Kelley Pace and R. Barry, “Sparse spatial autoregressions,” *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[77] C. Homburg, S. Kuester, and H. Krohmer, *Marketing management*. Springer, 2009.

[78] T. Trautner, F. Bolte, S. Stoppel, and S. Bruckner, “Sunspot plots: Model-based structure enhancement for dense scatter plots,” *Computer Graphics Forum*, vol. 39, no. 3, pp. 551–563, 2020.

[79] J. Jo, F. Vernier, P. Dragicevic, and J.-D. Fekete, “A declarative rendering model for multiclass density maps,” *IEEE Trans. on Vis. and Comput. Graphics*, vol. 25, no. 1, pp. 470–480, 2019.

[80] A. Abuthawabeh and M. Aupetit, “Toward an Interactive Voronoi Treemap for Manual Arrangement and Grouping,” in *EuroVis*, 2021, pp. 97–101.

[81] A. Abuthawabeh, A. Baggag, and M. Aupetit, “Augmented intelligence with interactive voronoi treemap for scalable grouping: a usage scenario with wearable data,” in *EuroVis*, 2022, pp. 43–47.

[82] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” in *The Craft of Information Visualization*, ser. Interactive Technologies, 2003, pp. 364–371.

[83] E. Gansner, Y. Koren, and S. North, “Topological fisheye views for visualizing large graphs,” *IEEE Trans. on Vis. and Comput. Graphics*, vol. 11, no. 4, pp. 457–468, 2005.

[84] C. Hurter, A. C. Telea, and O. Ersoy, “Moleview: An attribute and structure-based semantic lens for large element-based plots,” *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2600–2609, 2011.

Hyeon Jeon is a Ph.D. Student at the Department of Computer Science and Engineering, Seoul National University. His research interests span the field of Visual Analytics and Machine Learning. Before starting his Ph.D. program, he received a B.S. degree in Computer Science and Engineering from POSTECH.

Michaël Aupetit is a Senior Scientist at the Qatar Computing Research Institute. He graduated with a Computer Science Engineering degree and an MSc in Robotics and Microelectronics from the University of Montpellier. He earned a PhD degree in Industrial Engineering from Institut National Polytechnique de Grenoble (INPG) in 2001, and an HDR in CS from the University of Paris-Saclay in 2011.

Soohyun Lee is a Ph.D. Student at the Department of Computer Science and Engineering, Seoul National University. Before starting his Ph.D. program, he received a B.S. degree in Computer Science and Engineering from the Korea University, Seoul, Korea.

Kwon Ko is an incoming Ph.D. Student at Stanford University. Prior, he received a B.S. degree in Mathematics from Hanyang University and received an M.S. degree in Computer Science and Engineering from Seoul National University.

Youngtaek Kim is a staff engineer at Samsung Research. Prior to joining Samsung, we received a Ph.D. degree in Computer Science and Engineering from Seoul National University.

Ghulam Jilani Quadri received the PhD degree from the University of South Florida. He is currently an assistant professor at the University of Oklahoma. Before joining the University of Oklahoma, he was a CIFellow postdoc with the Department of Computer Science, University of North Carolina Chapel Hill.

Jinwook Seo is a professor in the Department of Computer Science and Engineering, Seoul National University, where he is also the Director of the Human-Computer Interaction Laboratory. His research interests include Human-Computer Interaction, Information Visualization, and Biomedical Informatics. He received his PhD in Computer Science from the University of Maryland at College Park in 2005.