Facilitating High-Dimensional Data Exploration with t-SNE Dynamics Visualization

Yingkai Song

Delft University of Technology

TU Delft

Facilitating High-Dimensional Data Exploration with t-SNE Dynamics Visualization

by

Yingkai Song

Instructor: T. Höllt
Faculty: Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

TU Delft

Summary

High-dimensional data are extensively generated and utilized across various fields. Dimensionality reduction techniques, such as t-SNE, create low-dimensional embeddings that are easier to visualize. Recent research suggests that the dynamics of the embeddings during t-SNE optimization can reveal valuable information. Building on this insight, we developed visualizations that enable efficient visual analytics of t-SNE dynamics, helping users derive insights more effectively. Preliminary evaluations indicate that our visualizations not only make tasks easier to perform with greater confidence but also have the potential to uncover additional insights.

Contents

1 Introduction

High-dimensional data is widely generated and utilized in fields such as finance, life sciences [29] [11] [48] [6], image analysis [31] [27], and retail [13]. Since high-dimensional data often contain valuable information, users seek to extract and visualize it in visual analytics [9] [25]. However, conventional visualization techniques, such as scatterplots, struggle with high-dimensional data due to human cognitive limitations in directly perceiving more than three dimensions.

Methods such as scatterplot matrices (SPLOMs) [7] [3] and parallel coordinate plots (PCPs) [24] provide partial solutions but encounter significant challenges with large datasets and high dimensionalities. PCPs, for example, do not scale well with increasing dimensions [18] and suffer from visual clutter due to overdrawn lines [16], while SPLOMs are effective only for a limited number of dimensions. Given that high-dimensional datasets often contain tens to thousands of dimensions, these visualization techniques become less effective [46].

To address the challenges, dimensionality reduction techniques such as Principal Component Analysis (PCA) [22], Uniform Manifold Approximation and Projection (UMAP) [30], and t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] are commonly employed. These methods transform high-dimensional data into two or three dimensions, making it possible to interpret the results using conventional visualization techniques like scatterplots. The resulting low-dimensional representations, known as embeddings, allow researchers to analyze intricate patterns within high-dimensional datasets. These dimensionality reduction techniques strive to preserve essential data characteristics, such as global or local structures, in low-dimensional embeddings, typically visualized as 2D scatterplots. As a result, analyzing these embeddings yields insights comparable to those derived from high-dimensional data. Among these techniques, t-SNE is particularly effective in preserving local structures, ensuring that similar points in high-dimensional space remain close in the low-dimensional embedding while dissimilar points are kept apart. This capability makes t-SNE a valuable tool for revealing relationships between data points, facilitating hypothesis generation and validation.

A recent study by Li et al. in single cell biology [29] extended the analysis beyond static t-SNE embeddings to explore their development throughout the optimization process. The authors discovered that the dynamic evolution of many clusters aligned with real cell differentiation pathways, revealed by cluster formation and neighborhood relationships over optimization iterations. This finding suggests that observing t-SNE dynamics can inform assumptions about cell differentiation behavior. To achieve this, users must generate snapshots (as shown in Figure 1.1) of representative embeddings throughout the t-SNE optimization process, summarizing key structural changes for further analysis. Li et al.'s study demonstrated the potential of visual analytics on t-SNE dynamics to provide deeper insights into high-dimensional data.

Despite the promising results of such studies, challenges remain in effectively visualizing and exploring the development of t-SNE embeddings. Current approaches either focus on precomputed final embeddings or dynamically compute and visualize embeddings on-the-fly, discarding intermediate embeddings and missing valuable insights. In Li et al.'s work, visualizations are limited to snapshots of scatterplots

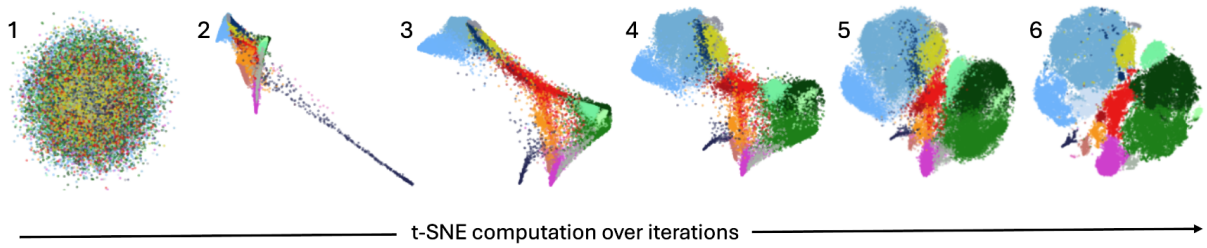


Figure 1.1: Typical start of visual analytics in a traditional approach. Users identify a few, in our case, six specific embeddings on the fly during the t-SNE optimization to represent stages in the optimization.

which failed to capture the movement of data points in detail and thus lacked a coherent narrative of the optimization dynamics. Furthermore, creating these snapshots manually is cumbersome and time-consuming. Integrating optimization dynamics into visualizations also introduces the risk of visual clutter, especially with large datasets. Some studies have attempted to visualize the sequences of data points across t-SNE embeddings to depict dynamics; however, these structures often differ significantly from t-SNE dynamics in terms of definition, size, visual element distribution, orientation, and direction, making them unsuitable without modification. These issues are further discussed in Chapter 3 and 4.

This research aims to enhance the exploration of t-SNE dynamics by allowing users to interactively explore how t-SNE embeddings evolve during optimization. We aim to provide users with concise, summarized visualizations that present key structures throughout the t-SNE optimization process. To achieve this, we introduce a visualization system that presents t-SNE optimization information from two perspectives while effectively managing visual clutter. While our system has been primarily tested on single-cell data using the t-SNE algorithm, it has the potential to be generalized to other high-dimensional datasets and dimensionality reduction techniques that iteratively optimize low-dimensional embeddings, such as UMAP. The key contributions of our work are as follows:

- We introduce a set of visual representations integrated into a visual analytics framework to facilitate the capture of structures during t-SNE optimization.
- We improve an existing bundling algorithm to reduce visual clutter while reducing information loss and preserving structures of t-SNE dynamics.
- We evaluate the effectiveness of our visualization through a pilot study with a domain expert, as well as a general study.

The remainder of the thesis is organized as follows: Chapter 2 introduces t-SNE; Chapter 3 summarizes the design requirements for the proposed visualization; Section 4 reviews related work; Section 5 discusses the design and implementation of our visualization system, including the visual representations and interactions; Section 6 presents an evaluation of the system's effectiveness, and Section 7 concludes the paper with limitations and potential directions for future research.

2

t-SNE

In this chapter, we introduce t-SNE [44], a fundamental technique for understanding related work, visualization requirements, and our contributions in the following chapters. Figure 2.1 illustrates a t-SNE embedding, revealing cluster structures that preserve high-dimensional neighborhood relationships in a 2D visualization. To achieve this, t-SNE iteratively optimizes a low-dimensional embedding using gradient descent on a cost function *C*, constructed with two kinds of similarities: the high-dimensional similarities represented by a symmetric joint probability distribution *P* calculated on every high-dimensional point pairs of the input data, and the low-dimensional similarities represented by a joint probability distribution *Q*, which are calculated in every iteration as the low-dimensional embedding updates.

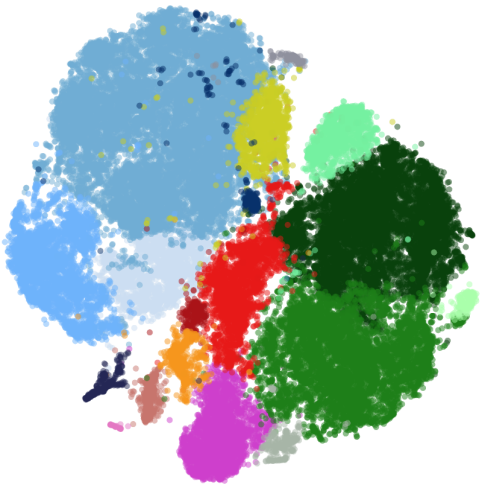


Figure 2.1: Example t-SNE Embedding. It is created with 44,789 immune cells sampled from human fetal intestine, after 1,000 iterations. The embedding is colored by clusters.

The cost function *C* in t-SNE is defined as the Kullback–Leibler (KL) divergence between joint probability distributions *P* and *Q*. As the optimization progresses and the similarities in the low-dimensional embedding increasingly reflect those of the high-dimensional data, the cost decreases. Given two data points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ in the high-dimensional data and $\boldsymbol{y}_i$, $\boldsymbol{y}_j$ in the low-dimensional embedding, their pair-wise similarities are represented by p_{ij} and q_{ij} respectively. The cost function is formulated as:

$$C(P,Q) = KL(P \parallel Q) = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} p_{ij} \ln \left(\frac{p_{ij}}{q_{ij}} \right) \tag{1}$$

To be specific, for each $\boldsymbol{x}_i$, a Gaussian kernel is centered on the it to compute the probabilities between the point and it's neighbors as their similarities. The symmetric joint probability p_{ij} is computed as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \tag{2}$$

where $p_{j|i}$ is a relative similarity between $\boldsymbol{x}_i$ and all its local neighbors $\boldsymbol{x}_j$. $p_{j|i}$ is represented as follows:

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i}^{N} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)} \tag{3}$$

The number of effective neighbors is decided by a key parameter called perplexity of value μ. It can be computed as follows:

$$\mu = 2^{-\sum_j^N p_{j|i} \log_2 p_{j|i}} \tag{4}$$

Perplexity balances local and global structure in the low-dimensional embedding: a smaller perplexity value focuses more on capturing the local structure (close neighbors), whereas a larger perplexity value accounts for both local and global structure, considering both nearby and distant points. σ_i is determined based on the perplexity value.

q_{ij} is calculated using a kernel of Student's t-Distribution with one degree of freedom instead of Gaussian for better quality of clusters formed:

$$q_{ij} = \left(1 + \|y_i - y_j\|^2\right)^{-1} Z^{-1} \tag{5}$$

where Z is the normalization term:

$$Z = \sum_{k=1}^{N} \sum_{\substack{l=1 \\ l \neq k}}^{N} \left(1 + \|y_k - y_l\|^2\right)^{-1} \tag{6}$$

The gradient of Kullback–Leibler divergence cost function C to low-dimensional embedding is given by:

$$\frac{\partial C}{\partial \boldsymbol{y_i}} = 4 \sum_{j}^{N} (p_{ij} - q_{ij}) \left(\boldsymbol{y_i} - \boldsymbol{y_j}\right) \left(1 + \|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2\right)^{-1} \tag{7}$$

$$= 4 \left(\sum_{j \neq i}^{N} p_{ij} q_{ij} Z \left(\boldsymbol{y_i} - \boldsymbol{y_j}\right) - \sum_{j \neq i}^{N} q_{ij}^2 Z \left(\boldsymbol{y_i} - \boldsymbol{y_j}\right) \right) \tag{8}$$

$$= 4 \left(F_{\mathrm{i}}^{attr} - F_{\mathrm{i}}^{rep}\right) \tag{9}$$

Here, the positive term of the gradient can be treated as attractive forces F_{i}^{attr} while the negative one as repulsive forces F_{i}^{rep}. In the early phase of t-SNE optimization, a technique called early exaggeration is used to amplify the attractive forces by multiplying q_{ij} by a constant number. The reason of doing this is that early in the optimization process, the points in the low-dimensional space are initialized randomly. Without early exaggeration, the attractive forces between points that are supposed to be close might not be strong enough to pull them together in a meaningful way, resulting in poor local minima where clusters are not well separated. After a predefined number of iterations, the exaggeration factor decays gradually until the exaggeration no longer exists. In later phases, points move further away at higher rates from the origin in the form of clusters as clusters repulse each other. Notably, throughout the optimization process, points tend to move from the origin towards the outside due to the repulsive forces being stronger than attractive forces, even during early exaggeration phase, albeit at a slower rate.

3

Requirement Analysis

In this chapter, we discuss the data to be visualized, user tasks, and requirements our visualizations should fulfill.

3.1. Data Definition

This study focuses on visualizing t-SNE optimization dynamics created during t-SNE optimization on high-dimensional data. Such data is created by storing embeddings of each t-SNE iteration instead of the final embedding alone. Specifically, we describe such data in a point-centric view: instead of a single low-dimensional position for each data point, a point of t-SNE dynamics is an *array* of low-dimensional positions representing positions at each iteration with order, since intermediate states of embeddings are stored instead of overwritten. Each data point can now be considered as a *trajectory* starting from the position of the first iteration to that of the latest iteration. In data visualization, such a trajectory with an origin (O) and a destination (D) is often referred to as a *trail* [41] [28], consisting of multiple connected straight-line segments. While the terms ”trajectory” and ”trail” are often interchangeable, we primarily use the term ”trajectory” but might switch to ”trail” when discussing bundling techniques in Chapter 5 due to its widespread use. This study specifically handles 2D data, though similar approaches can be applied to 1D and 3D. Such trajectory data, which captures the full history of t-SNE optimization, forms the foundation for our visualization and presents unique properties along with challenges, which will be discussed below.

3.2. Data Properties

Apart from the fact that such trajectory data differs from the t-SNE final embedding by definition, it also differs from many commonly visualized trajectories in the related work due to its high density and scale. The t-SNE trajectory dataset D contains I indices with an order ranging from 10^3 to 10^4, and the number of data points N can reach an order of 10^5 or 10^6. Consequently, visualizing the optimization process of t-SNE requires rendering large number of lengthy trajectories, necessitating careful visualization design to mitigate visual clutter for effective data exploration.

As discussed in Chapter 2, trajectories often originate near the origin and develop outward due to repulsive forces in t-SNE optimization, unlike geographic-related data where movements follow a mesh pattern instead of a radial expansion. Furthermore, since higher attractive forces in the earlier phase, points in an embedding can form clusters at an earlier iteration and tend to move collectively in subsequent iterations. Such movement patterns enable the application of visual clutter reduction techniques without directional constraints. Notably, trajectories made by connecting multiple *final embeddings*, although similar in data definition, do not share the same property, unless constraints on data initialization exist [38].

The force-directed analogy of t-SNE optimization means at each iteration, points experience different attractive and repulsive forces based on their current positions. Therefore, a point at a certain iteration can

move in any direction in the next iteration depending on the current forces. This leads to unpredictable movements and potentially zigzag patterns, especially when observed locally. These zigzag patterns introduce extra visual clutter but carry little significance for trajectory dynamics, as the focus is on overall global movement rather than local oscillations.

Since each point is affected differently by attractive and repulsive forces, movement varies in both direction and magnitude. In certain cases, points may end up occupying areas previously occupied by other points, especially if they belong to different clusters. This can result in trajectory overlap, as illustrated in Figure 3.1.

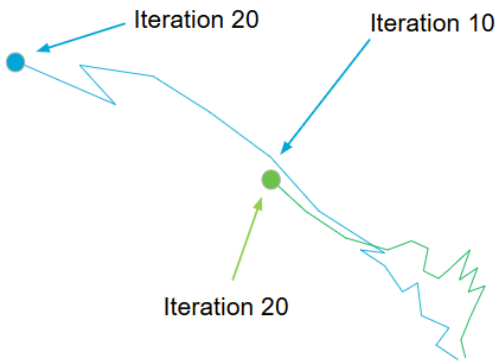


Figure 3.1: The green point at iteration 20 reaches where was occupied by the blue point at iteration 10

3.3. Task Abstraction

Visual analytics on both the standard t-SNE final embedding and t-SNE dynamics involve gaining an overview of the data, identifying clusters, and analyzing relationships between data points. Visual analytics of t-SNE dynamics specifically requires analyzing behavior and relationships over multiple iterations. Based on Li et al.'s work, we define the following key tasks:

- **T1: Browsing the Data Summary to Gain General Understanding**
  - **T1.1**: Understanding the overall structure of t-SNE trajectories.
  - **T1.2**: Navigating embedding positions over t-SNE iterations.
- **T2: Observing the Behavior of Points and Clusters Dynamically**
  - **T2.1**: Identifying which and when clusters split during t-SNE optimization.
  - **T2.2**: Determining the order of cluster splits over iterations.
  - **T2.3**: Analyzing relationships between clusters over iterations.

These tasks should be supported at any iteration of t-SNE optimization, and enable interactive exploration of data dynamics.

3.4. Requirements

Based on the problem analysis, data characteristics, and task abstractions, we define the following requirements (**V1 - V4**) that our visualization system should fulfill to effectively support visual analytics of t-SNE dynamics:

1. **V1** The visualization should be able to summarize the data in a single snapshot. This includes:
   (a) **V1.1** providing trajectory visualization as a foundation for all views and features; (**T1.1**)
   (b) **V1.2** providing navigation to a visualization snapshot for any chosen iteration(s). (**T1.2**)
2. **V2** The visualization should enable quick identification of cluster development over iterations. This includes:
   (a) **V2.1** providing trajectory representations independent of varying directions of point movements due to the constantly changing attractive and repulsive forces; (**T2**)
   (b) **V2.2** presenting the trajectories free from potentially varying scales in absolute positions due to the expanding nature of t-SNE. (**T2.2**, **T2.3**)
3. **V3** The visualization should scale well (**T1**, **T2**). This includes:
   (a) **V3.1** handling data with orders of 10^5 points in interactive time;
   (b) **V3.2** visualizing trajectories while minimizing visual clutter;

(c) **V3.3** balancing between information loss and visual clutter reduction.

4. **V4** The visualization should allow user-controlled data display (**T1**, **T2**). This includes:
   (a) **V4.1** allowing users to perform filtering on data display;
   (b) **V4.2** providing users the ability to control the coloring and appearance of data clusters of interest.

4

Related Work

As outlined in previous chapters, various efforts have been made to visualize t-SNE embeddings and trajectories in different forms. t-SNE visualizations are commonly employed in life sciences research to analyze high-dimensional data, such as tracing brain cell signals [11], distinguishing RNA sequences [48], and identifying species [6]. Most studies use only the final, converged embedding and present it in a 2D scatterplot (**V1.1**). Such visualizations are well-suited for research objectives in these research fields, mainly by displaying cluster structures, handle up to tens of thousands of data points (**V3.1**), and allowing some user-controlled data display (**V4**). However, they do not address other visualization requirements discussed in this study.

Li et al.'s research [29], as mentioned in Chapter 1, created a significantly different visualizations from t-SNE. Their work visualized a series of intermediate embeddings across optimization iterations. Their study demonstrated the value of visualizing t-SNE optimization process, revealing the actual development of cell differentiation. This approach presents a series of scatterplots illustrating embedding states throughout the optimization. However, instead of creating trajectories, this method overwrites intermediate embeddings as t-SNE evolves. Consequently, users must manually capture snapshots, preventing a comprehensive overview of t-SNE dynamics in a single snapshot (**V1**). Additionally, identifying cluster development is cumbersome due to the need for selection and browsing across multiple snapshots (**V2**).

As discussed in Chapter 3.1, the data we aim to visualize are trajectories. Various approaches have been developed to visualize trajectories in geographic data, such as traffic or migration flows. Early work by Tobler [43] used straight arrows to visualize origin-destination (OD) data, while Fadloun et al. [15] created trajectory visualizations for OD data to depict social network dynamics. These studies directly represent trajectories as connected line segments between 2D positions. Although this approach preserves 2D positional information, it can introduce visual clutter (**V3.2**) and lacks scalability for larger datasets (**V3.1**). When applied to tens or hundreds of trajectories, visual clutter becomes apparent, complicating structure identification of trajectories. Fadloun et al. provided basic selection mechanisms based on filtering conditions (**V4.1**) due to limited number of trajectories and lacked more flexible filtering methods such as brushing or display by clusters, as well as advanced coloring options(**V4.2**).

Based on working mechanism, trajectory visualization techniques can be categorized into three groups [41]: aggregation methods, density map methods, and edge bundling methods. The focus on these techniques are reducing visual clutter, while keeping essential information.

Aggregating methods simplify the trajectory data D into a reduced dataset D' with fewer trajectories. It reduces visual clutter by merging groups of trajectories into one. Figure 4.1 shows an example of hand-made aggregating method [32]. Automated methods are often related to calculating the proximity of certain areas as clusters and replacing OD points in these areas with usually centroid points, computed using clustering algorithms. For example, a refugee flow visualization [5] is proposed by only displaying the centroid origins or destinations with flow aggregation via hierarchical clustering; Phan et. al [37] and Guo [19] visualized various flow data using similar clustering variants. The drawing of D' can be done

with straight line fashion [5] if the number of trajectories are small, or generally with the help of graph drawing techniques to create less cluttered, curved drawings [37] [40]. However, the work discussed above usually assume pre-defined locations to build structures such as a graph for clustering. A variant of this approach is spatial abstraction techniques [1] [2], which similarly employs aggregated, length- and thickness-encoded arrows to represent trajectory structures. It differs in the way of generating the aggregation: it aims at trajectories with simply a series of numerical positions available and achieve the abstraction via Voronoi tessellation. While this method deals with similar type of data as ours and provides a simplified trajectory representation with good performance at multiple levels of detail, it shares similar issues with other aggregating methods. It abstracts the original spatial trajectory information, limiting user interaction with individual trajectories (**V4**) and loses level of details. (**V3.3**).

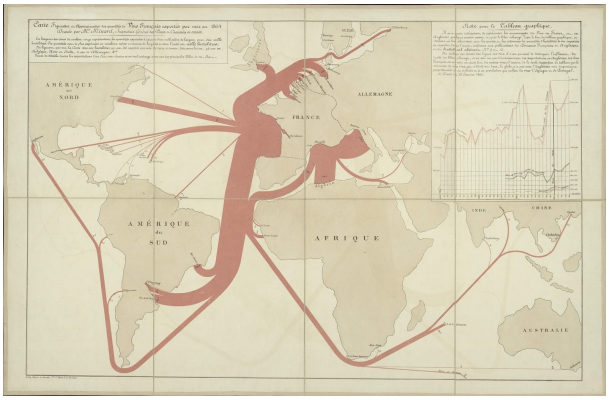


Figure 4.1: French wine export trajectories done in aggregating methods (Minard, 1864 [32])

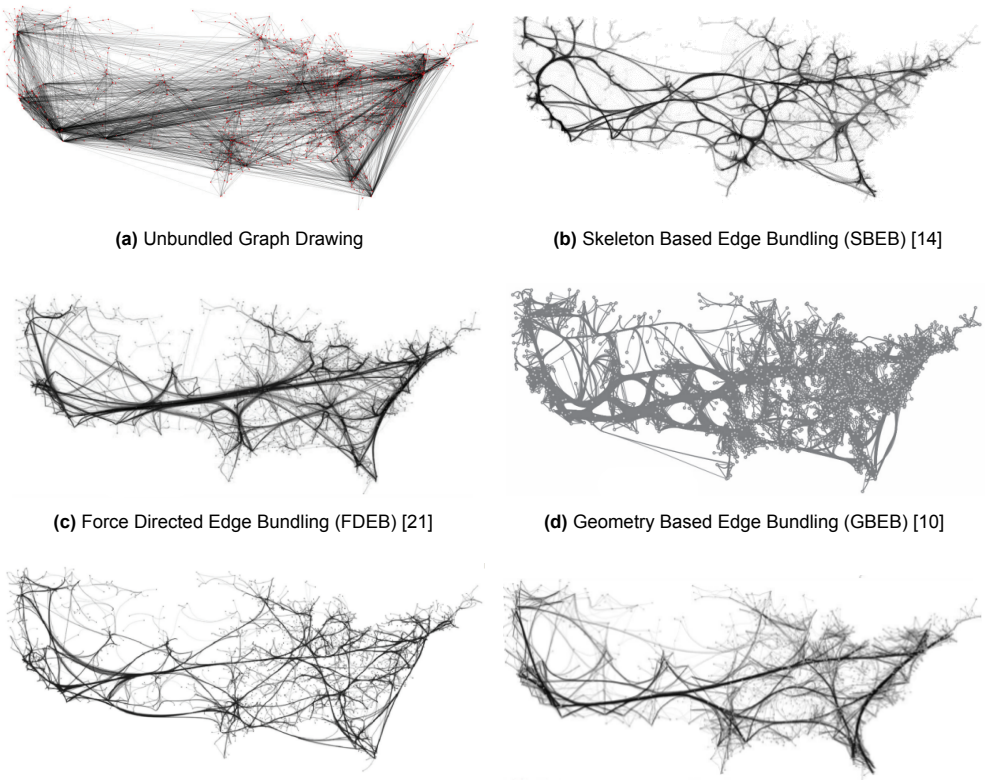
Density map methods address visual clutter by creating visualization using the *drawing* of trajectories, unlike aggregating methods which uses the (simplified) trajectory data. Such methods work by splatting a kernel for the drawing of D, producing a density map that provides insight into the distribution of trajectories, a process known as Kernel Density Estimation (KDE) [8]. Intuitively, such methods ”blur” the cluttered trajectory drawings into a less cluttered representation in image space. Density map methods effectively reduce local visual clutter since trajectories closer than the kernel size appear ”merged”, forming a single visual entity. However, as they visualize trajectory drawings in image space—similar to aggregation methods, users may find it difficult to interact with individual trajectories, as they are not explicitly present in the visualization.

Edge bundling techniques, in contrast, generate a new layout by grouping adjacent edges into denser clusters while maintaining separation through low-edge-density areas [28]. Unlike density maps, edge bundling does not remove or blur individual trajectories. This property makes edge bundling ideal for visualizations where frequent user interaction with trajectories is required. Various approaches have been proposed for graph drawing and trajectory data visualization [10] [26] [21] [17] [42] [14] [23] [45]. Figure 4.2 shows examples of edge bundling algorithms.

These visualization techniques prioritize clutter reduction, thereby enhancing the cluster structures within t-SNE trajectory data. When combined with t-SNE trajectories, edge bundling allows data filtering on the granularity of individual data item **(V4.1)**. Minimizing information loss **(V3.3)** is not typically a primary objective in these studies, as standard edge bundling does not preserve information of intermediate paths in trajectories. Rather, only the positional information of the origin and destination are faithfully kept. To address this, we propose an improved, data-driven edge bundling approach which will be discussed in Chapter 5.

Traditional geographic trajectories typically maintain a consistent range in 2D space over time. In contrast, t-SNE optimization trajectories expand as the optimization progresses, with distances between later iterations often being several times larger than those in the early stages. This expansion presents a challenge for trajectory navigation, especially when users need to examine earlier iterations. Therefore, corresponding visual design need to be adjusted to summarize the data and display to users clearly.

Rauber et al. [38] visualized trajectories in t-SNE space using edge bundling, in their research on the evolution of learned representations, or activations, within artificial neural networks. Their approach visualizes the evolution of neural network representations by generating a sequence of specially initialized t-SNE embeddings across hidden layers and training epochs, with edge bundling reducing visual clutter. A key distinction between their data and our approach is dataset size—roughly 2,000 points in theirs versus tens of thousands in ours. Additionally, while their trajectory paths typically contain fewer than 10 nodes, t-SNE optimization trajectories often involve hundreds. Using standard edge bundling algo-



(a) Unbundled Graph Drawing

(b) Skeleton Based Edge Bundling (SBEB) [14]

(c) Force Directed Edge Bundling (FDEB) [21]

(d) Geometry Based Edge Bundling (GBEB) [10]

(e) Winding Road Edge Bundling (WR) [26]

(f) Kernel Density Estimation Edge Bundling (KDEEB) [23]

Figure 4.2: Example Bundling Algorithms on US Migration Graph

rithms, their approach does not tackle loss of intermediate information **(V3.3)**. Moreover, the trajectories used in their work were already well-clustered as final t-SNE embeddings, , with the primary focus being the transition of *final* embeddings across different neural network representations. As a result, their method is not designed for identifying data dynamics in t-SNE development **(V2)**. Furthermore, their study does not discuss scalability **(V3)** with larger datasets or longer trajectories. To our best knowledge, no prior work has visualized lengthy t-SNE trajectories with larger size. In this study, we propose a data-driven edge bundling technique specifically designed for the visualization of extensive and longer t-SNE optimization trajectories, aiming to reveal t-SNE optimization dynamics more effectively.

5

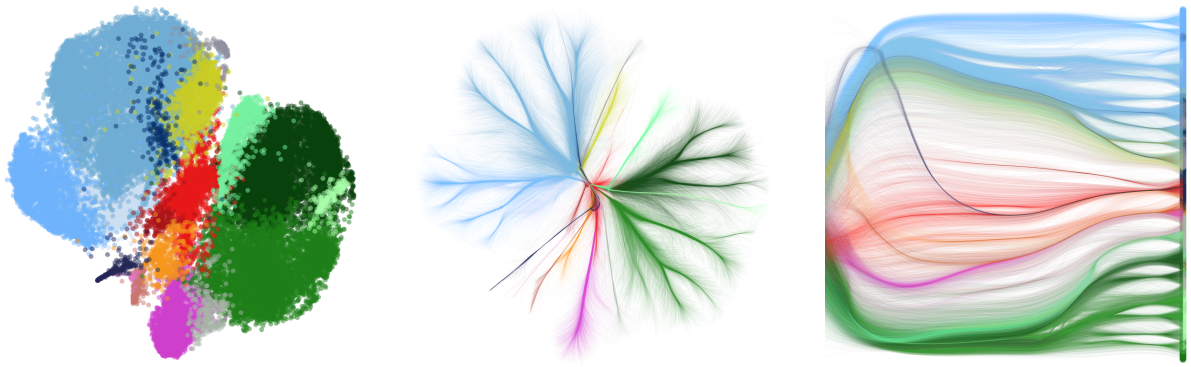
Visualization

In this chapter, we introduce the design and implementation of our visualization system, detailing the rationale behind the design choices. Our design is guided by user tasks, visualization requirements, and limitations identified in the related work, as discussed in Chapter 3 and 4. The visualization system was developed iteratively, starting with an early prototype that eventually led to the final design as a desktop application. We also discuss design alternatives that were considered during development.

Our solution aims to streamline the exploration process by integrating information into a single set of visualizations, eliminating the need to frequently switch between different embedding views. The design includes components of a 2D visualization, a 1D visualization, an adjusted image-based edge bundling algorithm to reduce visual clutter, coloring and a set of shared interactive components. These visualizations function independently or alongside traditional scatterplots. Figure 5.1 provides an overview of the visualizations.

5.1. Design

As discussed in Chapter 1, Li et al.'s workflow at the start of data exploration involves extracting several snapshots during the iterations. This is often done using existing visual analytics software (*e.g., Cytosplore* or *ManiVault Studio*). The process begins by running t-SNE on high-dimensional data, with real-time visualization of embeddings that update with each iteration. Users manually select and save snapshots of these visualized embeddings, often numbering in the hundreds, based on prior knowledge and experience, to create a sequence of embeddings called "stages". This process is considered the initial preparation before detailed data exploration can be performed and is usually cumbersome and time-consuming. We base our design on their workflow and try to introduce a more straightforward workflow and more informative visualizations.



(a) The Traditional Scatterplot Visualization **(b)** The 2D Visualization **(c)** The (normalized) 1D Visualization

Figure 5.1: Visualization Overview. All three visualizations are colored by clusters in this figure.

Li et al. used mass cytometry data of human fetal intestine innate immune cells, consisting of 44,789 data points with 36 marker dimensions. Li et al. observed 18 clusters in the dataset, each of corresponding to a specific type of cell, such as NK cells, int-ILC cells, ILC1, ILC2, ILC3 cells, etc. Some clusters represent subsets of the same cell type, distinguished by variations in marker expression. In the following sections, we discuss our visualization using the same dataset.

The visualization in Li et al.'s work has notable drawbacks: it does not summarize trajectory information (**V1.1**), and navigation over iterations in such solution is cumbersome (**V1.2**), hindering user task **T1** (Browsing the Data Summary to Gain General Understanding). Besides, the identification of cluster development is not easy due to the selection and browsing over the snapshots (**V2**), hindering user task **T2** (Observing the Behavior of Points and Clusters Dynamically). Applications with t-SNE typically target 2D visualization displayed on screen and the final converged embedding is visualized by showing its positions as points in a 2D scatterplot. In visual analytics software applications, intermediate steps of the embedding positions during the optimization are also visualized on the fly as they are computed. Based on this observation, an intuitive approach as an extension to current work to summarize trajectory information (**T1**), is to visualize the trajectory of each data point as it moves in the 2D space as a curve. To achieve this, we first visualize trajectories with both of the dimensions of embeddings as a faithful summary of t-SNE development, referred to as the 2D visualization.

5.1.1. 2D Visualization

Figure 5.2 shows a basic 2D visualization. For each data point, all 2D positions of the embedding are connected sequentially with line segments, displayed as trajectories in a single visualization, addressing **V1.1**. Users can highlight the final embedding positions on demand and adjust the iteration range for inspection (**V1.2**). This overview works as a basis of the following visual analytics and is useful as a starting point of the exploration.

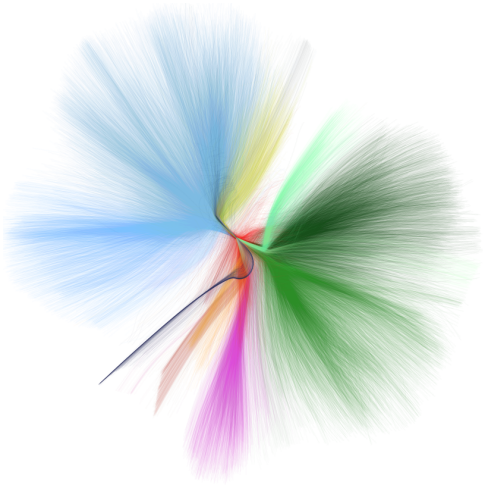
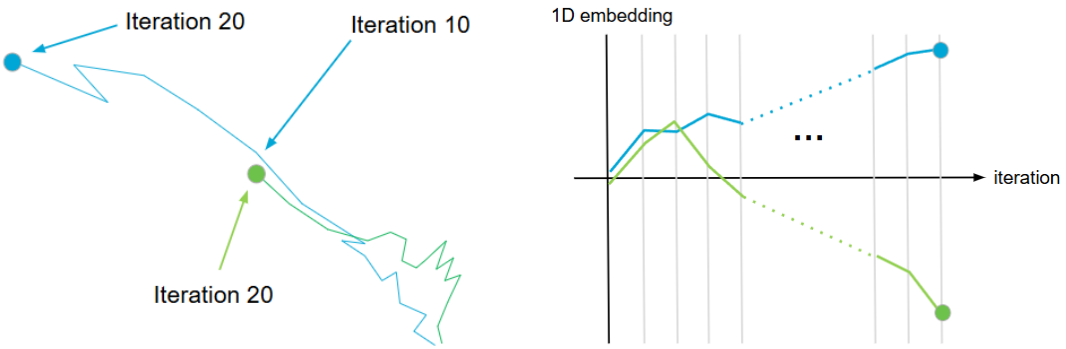


Figure 5.2: Basic 2D visualization

The 2D visualization effectively visualizes the positions of embedding points over t-SNE iterations with freedom to navigate, the 2D visualization presents challenges: once a 2D visualization is created for a certain iteration range, specifying embedding positions for a particular iteration is difficult. As discussed in the data properties, each trajectory consists of numerous straight path segments with varying lengths, potentially moving in arbitrary directions. This results in two key issues: First, curves that represent trajectories can overlap at any area in the 2D visualization and cause visual clutter; Second, the iterations are not explicitly encoded, making it difficult to track the dynamics of such trajectories. For example, users may struggle to determine whether splitting lines indicate actual cluster separation or if points merely coincide at different phases of optimization. Therefore, 2D visualization struggles to facilitate users to conduct task **T2** (Observing the Behavior of Points and Clusters Dynamically) although it helps conduct task **T1** (Browsing the Data Summary). Figure 5.3a demonstrates the second issue using two trajectories, each spanning 10 iterations. As shown, although the two trajectories are close in some line segments, these segments belong to different iterations, which means at any iteration during the optimization, the two trajectories are not close to each other.

Encoding iteration information into the 2D visualization straightforwardly mitigate these issues. However, we do not integrate opacity and line width iteration encoding into the 2D view, as this could introduce excessive visual clutter in datasets with many iterations and data points, violating requirement **V3.2**. Instead, we introduce another visualization that incorporates iteration information while working alongside the existing 2D visualization. One possible approach is to visualize all embeddings in a shared 2D space while introducing a third dimension for iterations, effectively creating a 3D visualization. However, we aim to maintain a purely 2D visualization. To achieve this, we explore methods to represent t-SNE positions in a single dimension while using iteration as a second dimension, forming a 2D visualization



(a) The green point at iteration 20 reaches where was occupied by the blue point at iteration 10

(b) In this example 1D visualization, it is easy to track the state of trajectories at each iteration.

Figure 5.3: From the 2D to 1D visualization

that enhances interpretability, as shown in Figure 5.3b.

5.1.2. 1D Visualization

To this end, we introduce a second visualization in addition to the 2D visualization, called the 1D visualization. Figure 5.4 provides an example of what it looks like. The core idea is to compress the higher-dimensional information into a single dimension, producing a 1D embedding. This representation is visualized as a line chart, where one axis represents iteration indices, and the other represents 1D embedding values, hence the term 1D visualization. Horizontally, points between embeddings always move from the left to the right at a same rate across iterates. This design allows users to compare relative positions between trajectories at the same iteration, reducing the aforementioned limitations of the 2D visualization. By eliminating distractions caused by varying movement directions and distances during t-SNE optimization, the 1D view enhances users' ability to track cluster evolution, fulfilling requirement **V2**. We considered two possible approaches: one is to directly transform the 2D embeddings in the trajectory data into 1D using space-filling curves; the other is to compute a 1D t-SNE embedding, independent from the existing 2D embeddings. We chose to use 1D t-SNE, the second approach. Below we discuss both approaches and the reasoning behind our design choice by analyzing their strengths and weaknesses.

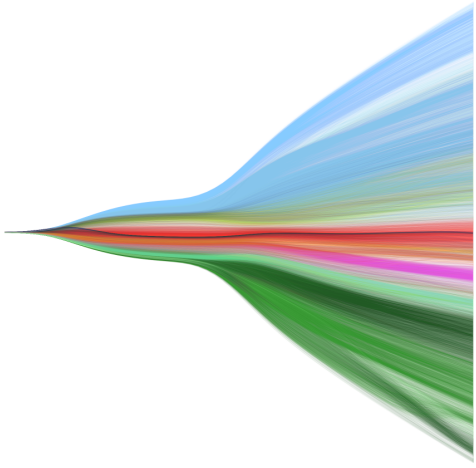
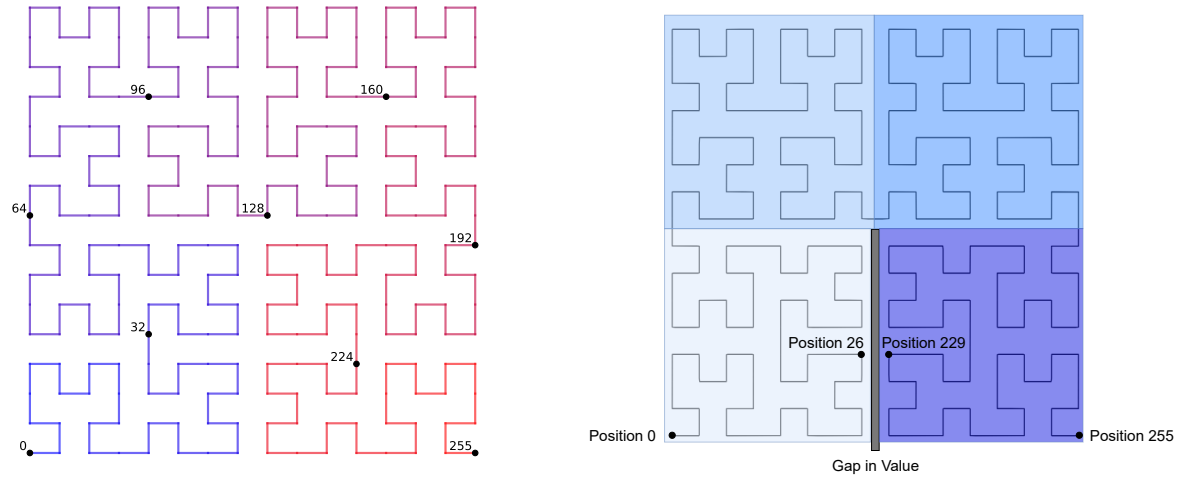


Figure 5.4: Example 1D Visualization

Space-filling Curves

A space-filling curve is a discrete curve with its range covering every position in a higher-dimensional space (e.g., a 2D unit square in our case) [39], providing a way to reduce dimensionality by indexing each element in the high-dimensional data to a specific position on the curve. Space-filling curves enables mapping the 2D t-SNE embedding onto a 1D space. Several space-filling curves exist, including the Hilbert curve [20], Peano curve [35], Morton curve [34] and Moore curve [33]. The Hilbert curve, for instance, maps discretized positions in 2D space to a 1D curve, as shown in Figure 5.5a.

Despite their locality-preserving properties, space-filling curves introduce distortions and inconsistency in some areas between 2D and 1D. Taking Figure 5.5b as an example. The curve are created as a 1D line with ordered indices covering the area of 2D space so that each position in 2D can be mapped onto a position in the 1D curve. We separate the 1D values into four parts, arranged from low to high and colored from light blue to dark blue. We notice that points mapped to position 26 and position 229 in the curve are close neighbors in the 2D space, but they belong to the areas of the lowest and highest val-



(a) An example of Hilbert curve with order 4. Indices are color encoded from blue for lower ones to red for higher ones.

(b) Drawback of space-filling curves. Taking Hilbert curve of order 4 as an example, similar for other types of space-filling curves.

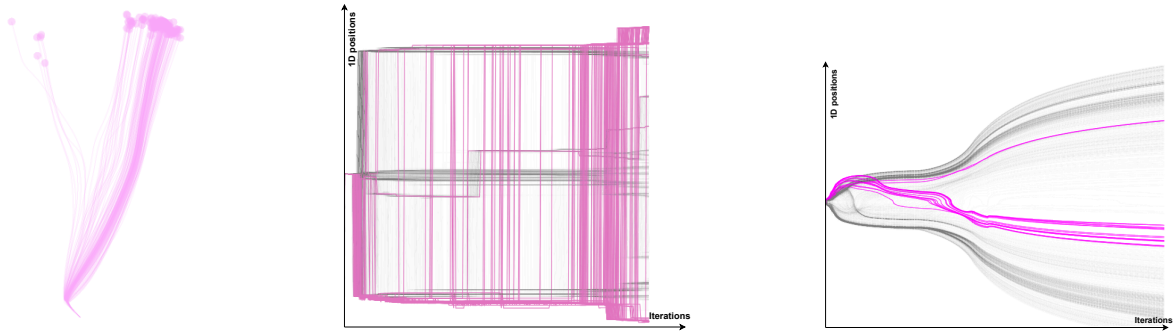
Figure 5.5: Comparison of Hilbert curve example and its drawback.

ued area respectively, placed distant in the 1D space. Such misalignment between 2D and 1D embeddings creates a gap in the visualization, resulting in discontinuity in 1D embedding over iterations.

Experiments confirmed that space-filling curves often introduce frequent jumps in the transformed 1D space, resulting in highly fragmented and cluttered visualizations, making it difficult to track t-SNE development, failing to fulfill **V3.2**. An example of the resulting 1D trajectory using a Hilbert curve with a group of data in pink is shown in Figure 5.6: The 2D scatterplot in Figure 5.6a shows the majority of data forms one cluster while a smaller minority forms another. In the 1D visualization using Hilbert Curve as embedding in Figure 5.6b, neighboring indices of positions jumps constantly and dramatically which hinders it from being useful. Moreover, due to the same reason, there could be points or even whole trajectories that are close together in the 2D embedding space but placed on different sides in the transformed 1D space. This drawback is inherent to all space-filling curves, as they may map two neighboring 2-dimensional positions into distant low-dimensional positions, thus breaking continuity and impairing visualization clarity. Due to these limitations, we discarded this approach and opted for 1D t-SNE.

1D t-SNE

Unlike space-filling curves, 1D t-SNE generates smooth and locally consistent trajectories. Figure 5.6c demonstrates that 1D t-SNE retains information of cluster development more effectively than space-



(a) **(b)** **(c)**

Figure 5.6: Choice of embeddings for the 1D visualization. Here we show only one group of data colored as magenta as example. Images here are taken from the our early prototype and trajectories in (b) and (c) are sub-sampled.

filling curves, preserving most relative positional relationships seen in the 2D visualization. 1D t-SNE is a variation of the standard t-SNE algorithm where the target dimension is set to 1 instead of 2. In this configuration, each high-dimensional point is represented by a 1D embedded point for each iteration. Unlike space-filling curves, 1D t-SNE always preserves local neighborhoods, ensuring that trajectories evolve gradually and coherently.

However, reducing the embedding to a single dimension comes with trade-offs. While 1D visualization aligns trajectories consistently across iterations in the x-axis, using one embedding dimension instead of two makes the visualization more compact, leading to less space to display structures compared to the 2D visualization. Therefore, we retain both 1D and 2D views in our system to provide complementary perspectives. A key challenge in combining 1D and 2D t-SNE is misalignment due to independent runs. Because 2D and 1D t-SNE use different initializations, the global positioning of embeddings may vary. For instance, a cluster that appears at the top of the 2D visualization might be placed at the bottom in the 1D visualization. However, since local structures are well preserved, this misalignment does not significantly impact data exploration. Alignment between 2D and 1D embeddings is out of the scope of the work. Recent work addressed this issue by modifying the t-SNE cost function, which will be discussed in Chapter 6.

Normalized Valued Embeddings

Another challenge in 1D visualization is the expansion of embedding sizes over iterations. Early iterations often have a much smaller value range than later ones. This results in earlier embeddings being compressed into a tiny portion of the visualization when displaying all iterations at once, reducing visibility. This limitation weakens the visualization's ability for users to conduct task **T2** (Observing the Behavior of Points and Clusters Dynamically). To address this, we apply normalization to embeddings at each iteration. It ensures that all iterations share a common value range, making data relationships more comparable over iterations. Figure 5.7 illustrates the effect of such normalization: embeddings from earlier iterations are expanded and rendered in a larger area on the screen, making it easier for analyzing dynamics. Because normalization only rescales values without altering relative distances, local neighborhood structures are preserved. Normalized embeddings enhance visibility across iterations, allowing users to track cluster evolution without excessive zooming. This approach improves **V2** (quick identification of cluster development) while also fulfilling **V1** (summarization) by providing a clearer global view of t-SNE optimization.

5.1.3. Edge Bundling

The aforementioned 1D and 2D visualizations do not explicitly address the issue of visual clutter (**V3.2**) caused by the occlusion of many overlapping trajectories. Unlike final t-SNE embeddings, where points are already well clustered, t-SNE trajectory visualizations include intermediate iterations before clusters have fully formed. Moreover, curves take up more space than points. As a result, trajectory-based visualizations often suffer from significant occlusion. Figure 5.8a shows a typical case of trajectory data

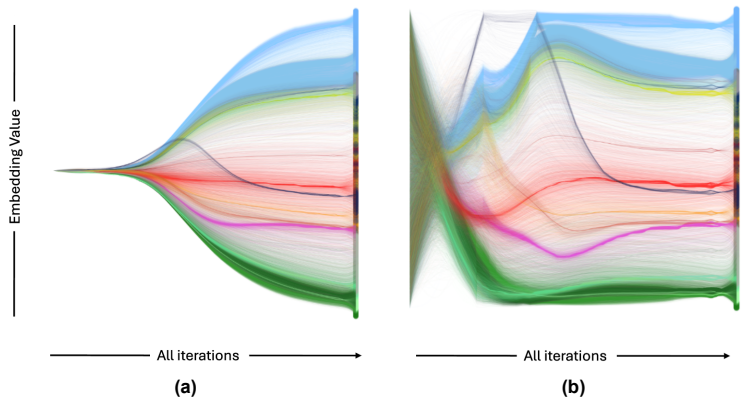
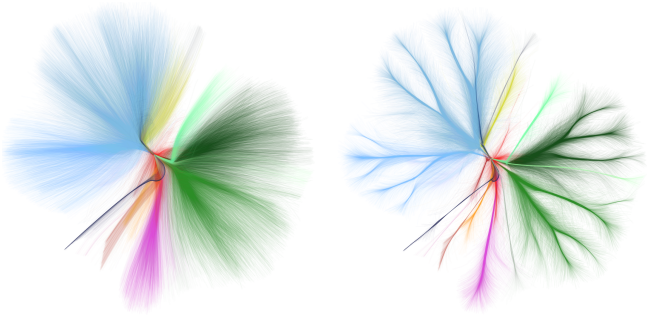


Figure 5.7: Comparing the 1D visualization without (a) and with (b) normalization

with visual clutter. First, certain clusters are not visible due to the dominance of larger clusters. In the figure, although a total of 18 classes are rendered, only around 12 of them are clearly visible. Clusters with fewer data points are hard to identify. For instance, the dark colored cluster and the gray cluster at the top, the grayish green cluster at the bottom, and the light green cluster on the right, are barely visible. Second, cluster evolution is difficult for users to analyze. This is because the trajectories are loosely presented in the 2D space, especially for larger clusters, making it hard to discern structured development patterns. To analyze the dynamics of clusters, users should be able to have access to visual representation that tightly organize similar trajectories into strands, preferably with controllable level of detail. To handle this, we introduce an bundling method (KDEEB) and apply it on trajectories in both the 1D and 2D visualization. The result of this bundling is shown in Figure 5.8b, which shows both clear presence of all clusters and the main structures of cluster development. We eventually modified this algorithm to suit t-SNE trajectory data, which will be discussed later in this section.



(a) Unbundled Trajectories **(b)** Bundled Trajectories

Figure 5.8: Visual Clutter: Issue and Solution with Bundling Techniques

Edge bundling (sometimes referred to simply as bundling) reorganizes the drawing of trails, creating visually smooth and compact representations within a limited drawing space. Most edge bundling algorithms modify path endpoints in trails, except for the origin (O) and destination (D). The key requirements of edge bundling algorithms for our visualizations are:

- **Performance**: The edge bundling algorithm should compute the bundles within interactive time;
- **Compatibility**: The algorithm should run on the hardware used by target users without requiring specialized frameworks like CUDA.

Based on these requirements, we selected the kernel density estimation edge bundling (KDEEB) algorithm, which computes in seconds on dataset of order 10^5, runs on both CPU and GPU, and does not depend on specialized frameworks, ensuring compatibility with our target users' hardware.

KDEEB bundles edges using an image-based, density-driven approach. Formally, it creates simplified graph drawings [28] by applying the mean shift aggregation principle [8]. KDEEB estimates the density of points forming curves and uses it to pull points toward local high-density centers, resulting in a new representation of curves. The algorithm achieves this through five steps: edge resampling, density map computation by splatting, gradient estimation, edge advection and smoothing, as shown in Figure 5.9. These steps are done with multiple iterations to increasingly tighten the bundles. KDEEB is parameter-sensitive, and key parameters are discussed below alongside the main steps.

KDEEB needs to create a density field based on lines. As the density is estimated using kernel density method, which means kernels will be applied to points, the curves need to be resampled to create a set of sample points properly spaced to ensure the best results. The edge resampling step ensures that the

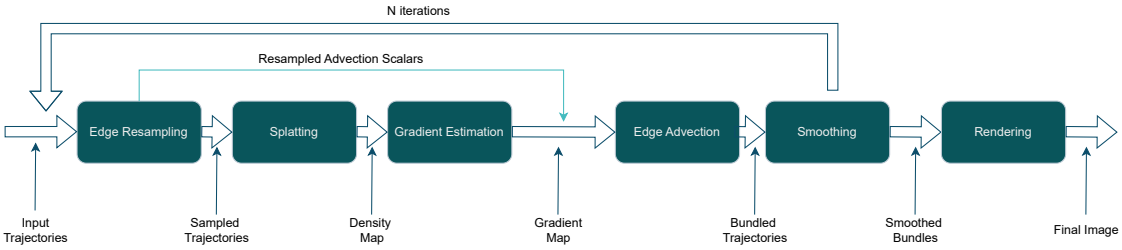


Figure 5.9: KDEEB pipeline

number of points along the trail is sufficient to create smooth curves, while also limiting the number of samples to prevent performance issues and visual artifacts. Two sampling threshold parameters control how many sample points will be created: split distance and remove distance. If two neighboring sample points are too far apart, such that their distance exceeds the split distance, a new sample point is inserted at the midpoint of the segment, splitting the line segment. Conversely, if two neighboring sample points are too close, such that their distance is less than the remove distance, one of the points is removed. In the normalized 2D space of $[-1, 1]$, the selection of these thresholds primarily depends on the overall number of points along a trail. More points will lead to lower thresholds to retain the positional information of trajectories.

KDEEB creates a density map by splatting a square kernel onto each sample point, accumulating values in the pixel space in an off-screen buffer, and storing the result as a texture. The width of the kernel controls the extent of its influence; a larger kernel will have a broader effect, causing more points to contribute to the overall density map and vice versa. Common kernel types include the circular linear gradient kernel, Gaussian kernel, and Epanechnikov kernel. For each dimension of the kernel, we assume it has a width of $2w + 1, w \in N^+$ and kernels are sampled in the range $[-w, w]$. For ease of comparison, we define all 3 kernels in such a way that they all give value 1 when evaluate at origin. The circular linear gradient kernel is a cone-shaped kernel with value decrease linearly from the origin to both direction of the axis and of value 0 at w or $-w$. It is defined as follows:

$$y = -\left|-\frac{1}{w}x\right| + 1 \quad (-w \leq x \leq w,\ x \in N) \tag{2}$$

Gaussian kernel used here has a curve peak of value 1, same as the other kernels. Since Gaussian kernel ranges to infinity, here we approximate its cumulative probability in the range of $[-3\sigma, 3\sigma]$ and map $[-w, w]$ to this range, which means for instance, the density at position w will be equal to that at 3σ. Therefore Gaussian kernel is defined as:

$$y = exp(-\frac{18x^2}{w^2}) \quad (-w \leq x \leq w,\ x \in N) \tag{3}$$

Epanechnikov kernel is a quadratic kernel, defined as:

$$y\ =\ 1 - \frac{1}{w^2}x^2 \quad (-w \leq x \leq w,\ x \in N) \tag{4}$$

For circular linear gradient kernel and Epanechnikov kernel, the value at both sides in each dimension is 0, while Gaussian gives close-to-zero values. The type and size of kernels determine the shape of the density signal contributed by a single sample point, resulting in density maps with varying features. Figure 5.10 illustrates these three kernels, and Figure 5.11 compares density map created with these kernels with varying kernel sizes. As shown in the first and third row of Figure 5.11, circular linear gradient kernel and Epanechnikov kernel produce similar density maps, leading to similar bundled results. For the Gaussian kernel covering $[-3\sigma, 3\sigma]$, the resulting density maps shown in the second row are slightly narrower in high-density area, leading to sharper local maxima and minima in the density map, more pronounced gradients and thus producing tighter bundles. However, such result is dependent on how we map $[-w, w]$ to the area in Gaussian kernel. Figure 5.12 shows the appearances of Gaussian kernels with varying range of mapping, density maps and bundled results they produce. The comparison between the first and second row reveals that mapping $[-w, w]$ to a wider range in the domain of Gaussian kernel results in tighter bundle due to thinner tails and vice versa.

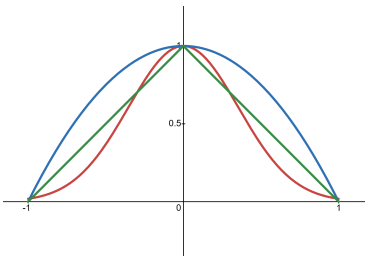
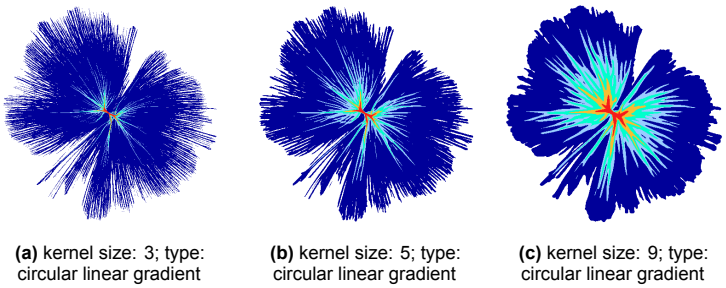


Figure 5.10: Typical Kernels, with the circular gradient, Gaussian and Epanechnikov kernels colored in green, red, and blue. Kernel sizes are 3. (figure made at https://www.desmos.com/)

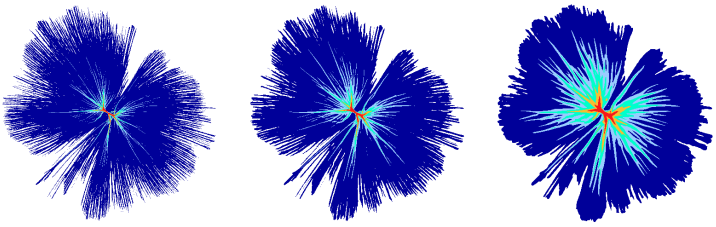
Additionally, density resolution, along with kernel size is a parameter in image pixel space that controls the granularity of the density map. While a higher resolution can significantly enhance the quality of the bundled result with less information loss, it also increases computational cost. The advection factor operates in the normalized t-SNE embedding space, controlling the speed of movement toward high-



(a) kernel size: 3; type: circular linear gradient

(b) kernel size: 5; type: circular linear gradient

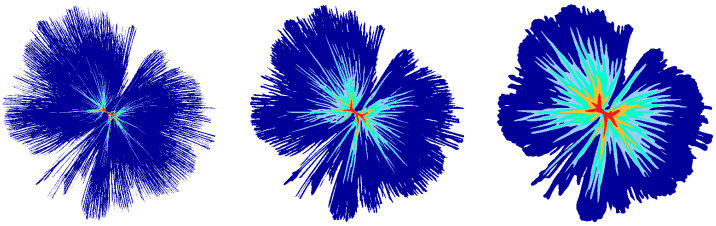
(c) kernel size: 9; type: circular linear gradient



(d) kernel size: 3; type: Gaussian

(e) kernel size: 5; type: Gaussian

(f) kernel size: 9; type: Gaussian

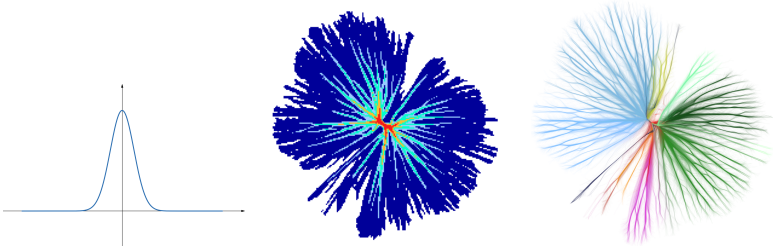


(g) kernel size: 3; type: Epanechnikov

(h) kernel size: 5; type: Epanechnikov

(i) kernel size: 9; type: Epanechnikov

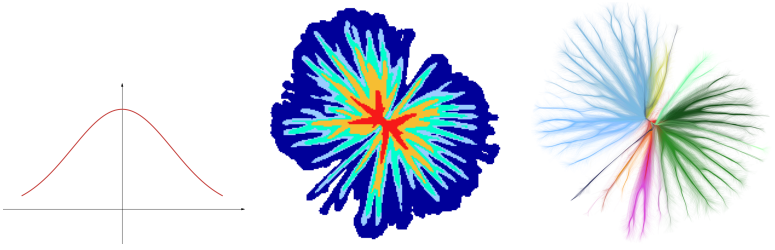
Figure 5.11: Density Maps with Varying Kernel Types and Sizes. Color map from low to hight density: dark blue, sky blue, cyan, yellow, red.



(a) Gaussian kernel covering [-4σ, 4σ]

(b) Density map using kernel (a)

(c) Bundled result using kernel (a)

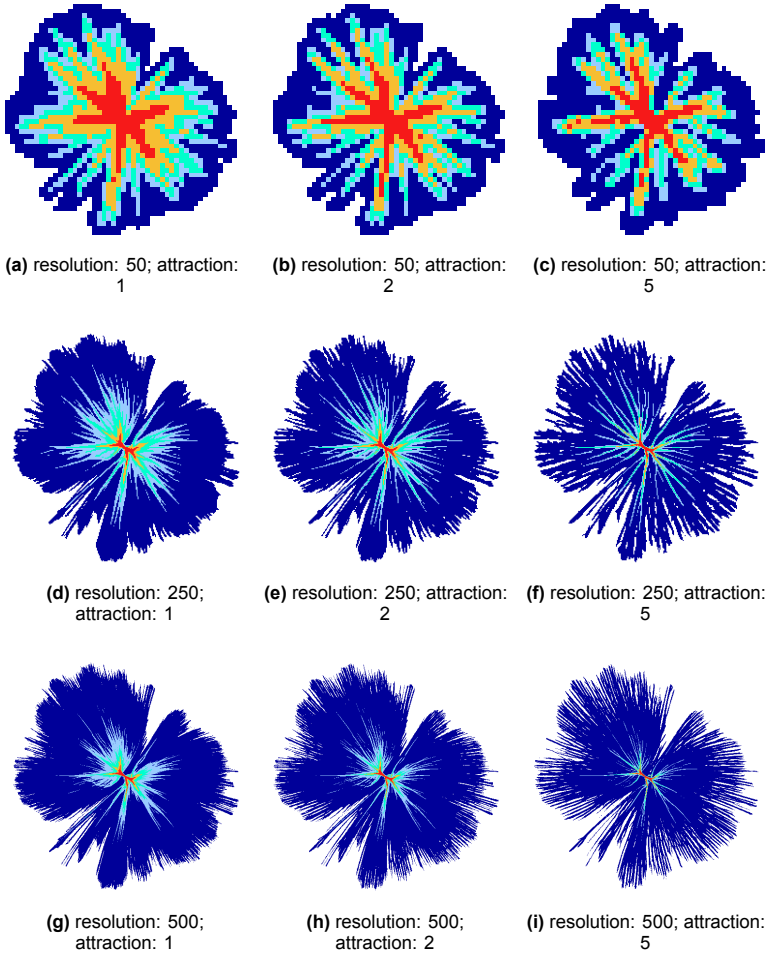


(d) Gaussian kernel covering [-σ, σ]

(e) Density map using kernel (d)

(f) Bundled result using kernel (d)

Figure 5.12: Comparison Between Different Mapping Range of Gaussian Kernel. For each kernel, its range in x-axis is $[-w, w], w \in N$; For both cases, resolution for the density map is 300, advection factor is 3 and iteration is 5.



(a) resolution: 50; attraction: 1 **(b)** resolution: 50; attraction: 2 **(c)** resolution: 50; attraction: 5

(d) resolution: 250; attraction: 1 **(e)** resolution: 250; attraction: 2 **(f)** resolution: 250; attraction: 5

(g) resolution: 500; attraction: 1 **(h)** resolution: 500; attraction: 2 **(i)** resolution: 500; attraction: 5

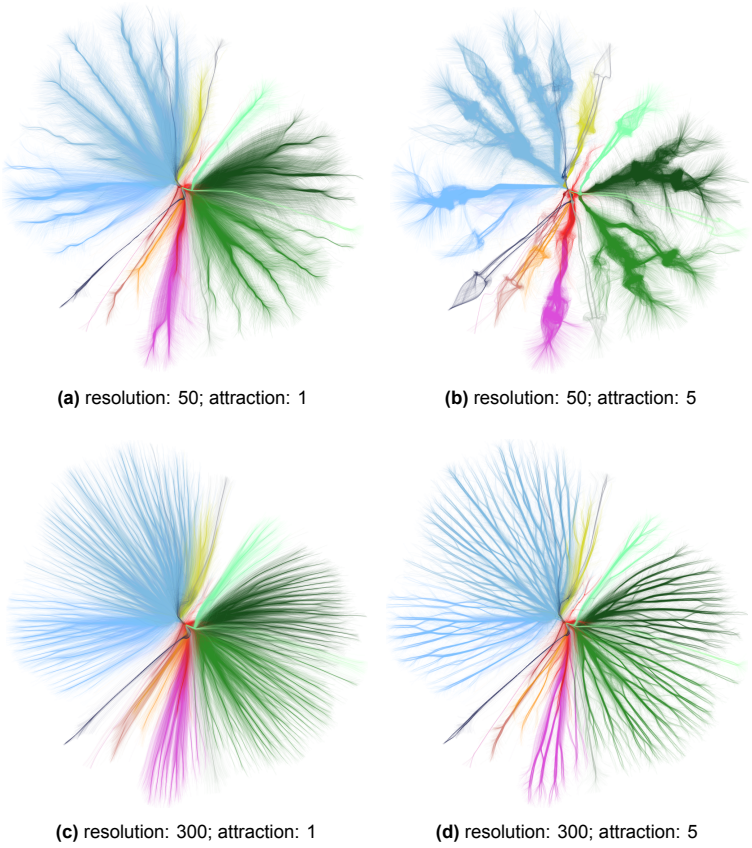
Figure 5.13: Comparison of Bundled Trajectories with Varying resolution and attraction factors. From left to right, attraction factors are 1, 2 and 3 respectively. From top to bottom, the resolutions are 50, 100 and 200 respectively. As are shown here, larger resolutions match with larger attraction factors to produce tightly bundled results with less artifacts and vice versa. Besides, higher resolution means sample points are processed in finer granularity, leading to more iterations before convergence. Here, for resolution 50, 100 and 200, we perform 5, 7, 10 iterations respectively.

density areas. Figure 5.13 compares results when changing resolutions and attraction factors. A higher resolution brings density maps of higher quality and higher attraction factor creates more profound local high-density areas, and vice versa. All images are generated after 5 iterations. As the kernel size is set constant in these examples, we observe the area of influence of each kernel reduces when the resolution increases.

For sufficiently- but not overly-bundled results, a large resolution should be paired with a large attraction factor and vice versa. A lower resolution combined with a higher attraction factor gives ”over-fitted” bundled with points pulled across the ridge of the high-density area, while a higher resolution combined with a higher attraction factor does not provide well-bundled results, as shown in Figure 5.14b and Figure 5.14c. Figure 5.14a and Figure 5.14d show ideally-bundled results with proper combination of the two parameters.

Next, the gradient of the density map is estimated for each position in the density grid using the density values and relative positions of its neighbors. The gradients are then used to update point positions in the subsequent edge advection step, where points are drawn toward areas of higher density. The advection speed is a key parameter influencing this process and will be discussed in detail later.

To push each sample point toward higher density areas, we compute the gradient $\nabla\rho$ of the density map at each sample point. This gradient is computed using a weighted sum approach, which approximates $\nabla\rho$ by considering neighboring density values within a square grid centered on each point, using spatial



(a) resolution: 50; attraction: 1 **(b)** resolution: 50; attraction: 5

(c) resolution: 300; attraction: 1 **(d)** resolution: 300; attraction: 5

Figure 5.14: Bundled Results from Combinations of Resolution and Attraction Factor

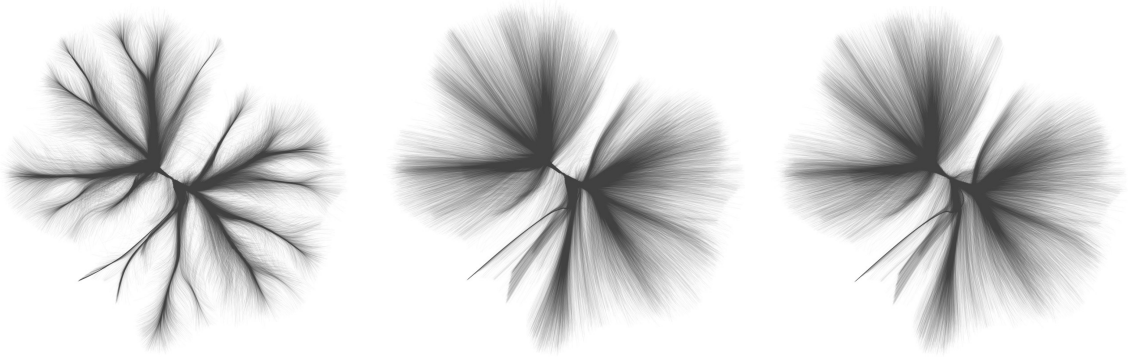
offsets dX and dY which are defined as the distances to the grid center horizontally and vertically. The gradient components $\nabla\rho_x$ and $\nabla\rho_y$ are computed as follows:

1. **Neighborhood Selection**: Each point's neighborhood is defined by a $W \times W$ grid window where W denotes the grid width as an positive odd number,.
2. **Weighted Accumulation of Density**: For each valid neighbor within the window, we calculate its distance to the central point along x and y directions, denoted dX and dY. Each neighbor's density value contributes to the gradient based on $||dX||$ and $||dY||$, effectively assigning a lower influence to closer neighbors to reduce point oscillations or even crossing the density maxima abruptly. The computations are written as follows:

$$\nabla\rho_x = \sum_{i,j\in\text{grid}} \rho(i,j) \cdot dX$$

$$\nabla\rho_y = \sum_{i,j\in\text{grid}} \rho(i,j) \cdot dY$$

Normalization is performed on each gradient to ensure smooth movements of sample points between iterations. An alternative approach is to normalize the density map instead of normalizing each gradient. This approach preserves the magnitude of gradients more faithfully but it does not lead to better bundling quality. Since t-SNE trajectories start in a small area near the origin, they are organized more densely in earlier iterations than in later ones. Ideally, the bundling power applied on these compact trajectories at earlier iterations should be made weaker to avoid artifacts. However, the area corresponding to earlier iterations are the most likely to be of high densities. Normalizing the density map instead of the gradients, can easily break the relationships between trajectories. Scaling the gradients to reduce movements in the high-density area will in turn make the bundling effect of the remaining area too weak, as shown in



(a) Normalized gradient **(b)** Normalized density map **(c)** Unbundled trajectories (reference)

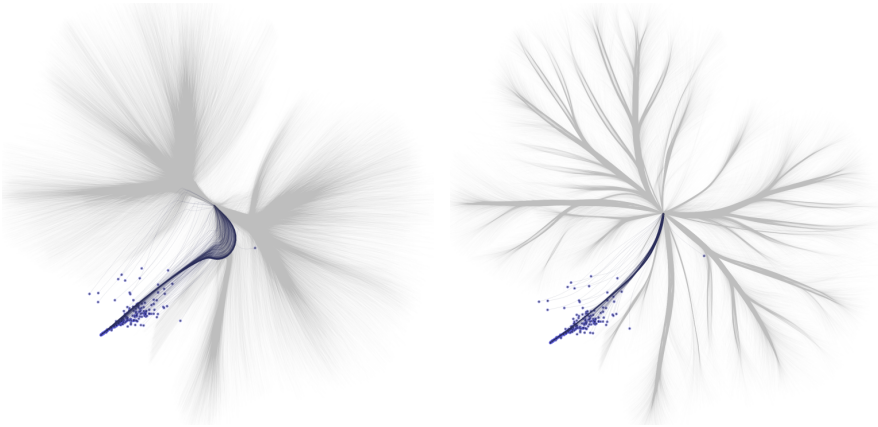
Figure 5.15: Comparison between two different gradient processing approaches

Figure 5.15b. As a result, we take the first approach for a consistent step size, as shown in Figure 5.15a. To update sample point positions, the product of the gradient and a user-defined scalar, which controls absolute bundling power, is added to the previous point position. So far, the bundling power mainly depends on the gradient. We will discuss other parameters that make edge advection more data-driven later in this section.

Finally, to remove artifacts in the results of edge advection and ensure more stable results across iterations, a Laplacian smoothing step is applied. Typically, 5..10 smoothing iterations can produce desired result [23]. The full process of density map calculation, gradient estimation, and edge advection is performed on the GPU, with the results stored as textures that are processed similarly as image data.

Edge bundling is more effective for visualizing lines with sparse intermediate points, such as airline routes or graphs in abstract spaces where intermediate positions are less critical than overall origin-destination information. However, for t-SNE trajectories, despite existing in an abstract space, intermediate point positions hold significance. Since edge bundling algorithms typically do not examine trajectory structures before updating sample point positions, applying standard edge bundling to such data can lead to a significant information loss. Figure 5.16 illustrates this issue, showing the loss of trajectory details in the dark blue cluster when using basic KDEEB. The early change of directions in the trajectories become obscured as the bundling process iteratively pulls them toward centers of density, eliminating critical structural details. In our work, we need to balance clutter reduction (the extent of bundle tightness) with the preservation of key positional information in trajectories **(V3.3)**.

A general approach to preserving trajectory information while reducing visual clutter involves segmenting trajectories and fixing the start and endpoint of each segment. A naive approach is to select fixed points



(a) Unbundled trajectories **(b)** Basic KDEEB bundles

Figure 5.16: Information loss caused by basic KDEEB

using equally spaced samples. However, our experiments indicate that this method introduces heavy visual artifacts around the fixed points, creating discontinuities in segment connections. Meanwhile, segments containing directional information may still be distorted if the these informative sample points do not align with the fixed points.

A more effective approach selects fixed points in a data driven manner. We propose an adaptation of KDEEB that leverages specific trajectory attributes to control bundling. The goal is to minimize movements at key positions, such as turning points, while allowing more aggressive bundling to less critical regions. In practice, we assign small scalar on bundling power for points with certain attributes instead of completely fixing their positions. We focus on two attributes: the turning angle between neighboring trajectory segments and the iteration stage of a given sample point within the trail.

- **Turning Angle**: During t-SNE optimization, cluster splits often manifest as groups of points diverging from a previously inseparable cluster, causing noticeable curvature changes in trajectories over multiple iterations. In contrast, relatively straight movements typically indicate different dynamics, such as points moving away from other distinct clusters or moving with other non-separable clusters before separation. We reduce bundling power on segments with large turning angles, enabling users to discern such movements in a decluttered yet informative manner (**V3.3**).
- **Iteration Stage**: As t-SNE optimization progresses, points are gradually pushed away from the origin in all directions within the 2D space, increasing pairwise distances between points. Standard KDEEB applies a uniform advection speed across all iterations. When applied to t-SNE trajectories, this can cause a misrepresentation of cluster-wise relative positions. An advection speed suitable for later iterations (with larger embedding values) might cause the bundled positions from early iterations to overshoot, akin to using an excessively large step size in gradient descent. Conversely, lower advection speeds may result in inadequate bundling for later iterations, limiting clutter reduction.

Our modified KDEEB approach adjusts advection speed dynamically based on trajectory attributes to address these issues. This modification is illustrated in Figure 5.17. For each point within trajectories, excluding start and endpoints, we compute the angle between its previous and next positions. Based on this angle, we assign the advection speed $s_{n,i}$ for the nth trajectory at iteration i within the range $[s_{min}, s_{max}]$, where s_{min} corresponds to an angle of π and s_{max} corresponds to an angle of 0. Next, $s_{n,i}$ is adjusted linearly based on the iteration index i, such that $s_{n,i} = s_{n,i}/\text{scalar}$ for $i = 1$, and $s_{n,i} = s_{n,i}$ for $i = I$, where scalar > 1. An example of two cases of small and large turning angles is shown in Figure 5.18 where we both consider the data position at iteration t. In Figure 5.18a, no change of direction happens, resulting in an advection speed $s_{n,i}$ with value s_{max}; On the other hand, in Figure 5.18b the change of direction is almost π, resulting in an advection speed close to s_{min}, meaning this point will be moved less by the edge bundling algorithm. Besides, the positions of the first few iterations are kept fixed, as the initial t-SNE embeddings typically lack meaningful structure. Advection speeds are interpo-

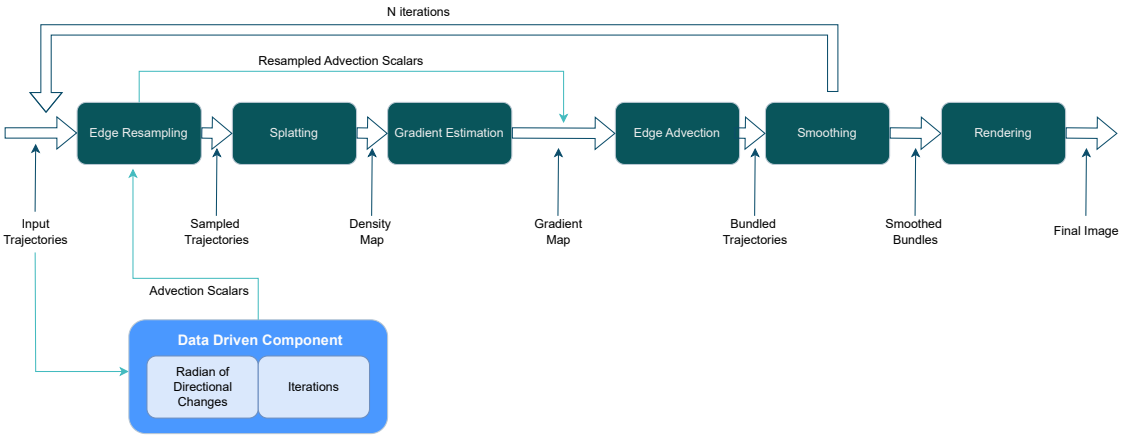
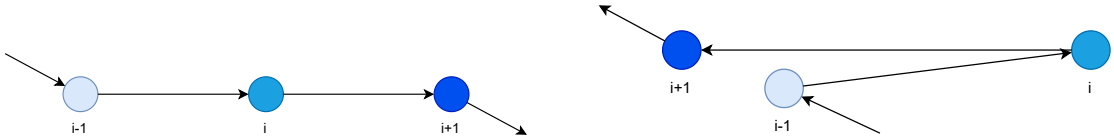


Figure 5.17: Data driven KDEEB made up of generic KDEEB pipeline at the top and the data driven component at the bottom that controls the amount of movement of each point based on the turning angle and iteration.



(a) 0 turning angle

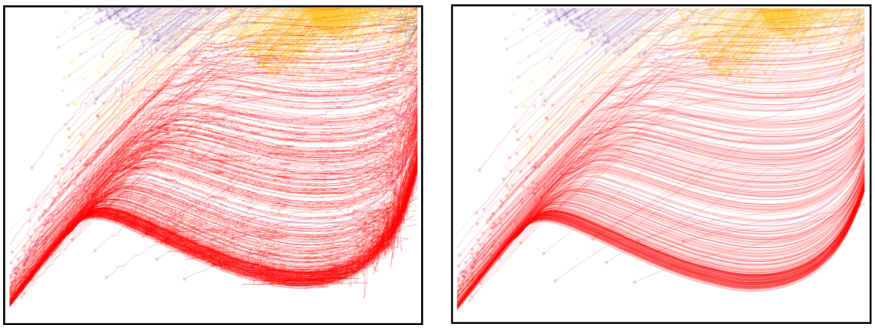
(b) A large (close to π) turning angle

Figure 5.18: Paths with different turning angles

lated during the edge resampling step and assigned to all sample points. As discussed in Chapter 3, t-SNE optimization follows a force-directed analogy, producing zigzag trajectories that introduce noise into turning angle calculations. To mitigate this, we apply a smoothing step before performing edge bundling, as shown in Figure 5.19. For trajectories with turning angles larger than a threshold, indicating significant structural change, we skip these points during the Laplacian smoothing step to avoid its position from being averaged by its neighbors. These points are identified before edge bundling and retained in the edge resampling step. Points that split next to these points, are also free from being smoothed. During data exploration, users can perform edge bundling at any desired iteration.

Figure 5.20 compares the results made by the improved KDEEB to the basic version. We can tell the darker cluster at the bottom goes in between the magenta and the green cluster in earlier iterations and was constantly changing moving direction during the period, from improved KDEEB in Figure 5.20c while basic KDEEB Figure 5.20a tend to erase such structures. Other examples are the light green and the dark green cluster on the right, as well as the darker blue and the yellow cluster at the top. On the other hand, since we aim to minimize movement in these structures during the bundling process, we omit the smoothing operation for sample points with large turning angles. As a result, some artifacts appear around the trajectory 'turning points', see the 'knot' looking structures in Figure 5.20c, especially when the cluster is sparse with a small number of trajectories, see the dark red cluster near the origin in Figure 5.20c. Such artifacts can be mitigated by applying additional weighted smoothing operations after the final bundling iteration. However, this comes at the cost of preserving the aforementioned structures. Therefore, we leave sample points with large turning angles unprocessed with Laplacian smoothing.

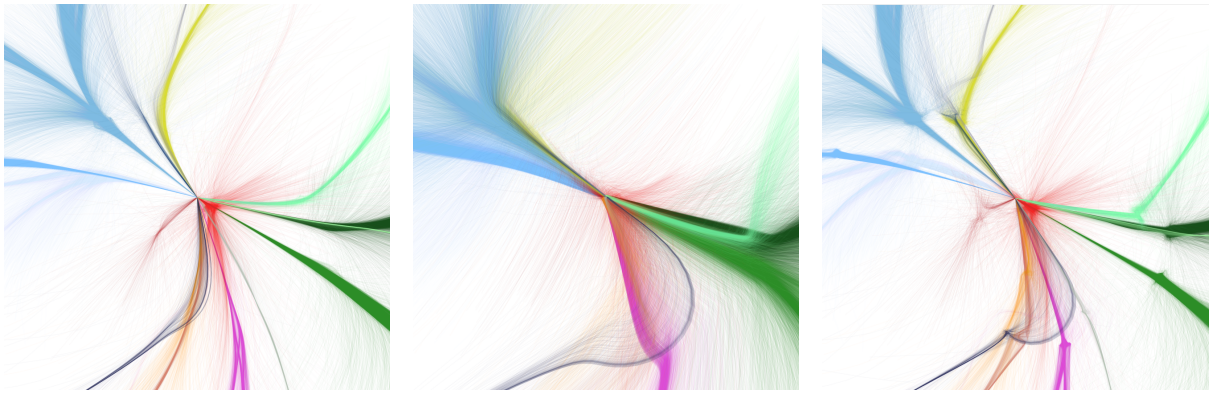
Edge bundling can be applied to all trajectories collectively or in a class-based manner, where trajectories within each cluster are bundled separately. In this approach, a separate density map is created for each cluster. Such class-based edge bundling can effectively separate bundles when clusters are compactly present in the 2D or 1D visualization. As shown in Figure 5.21, we observe separation between the yellow and gray clusters at the top left, the dark green and light green clusters in the mid left, and multiple colored clusters at the bottom left for class-based edge bundling, while standard one merges them during bundling as shown on the right. Users may select either one on demand: generally, class-based bundling provides more precise bundled results on each cluster, while the standard bundling provides better visual clarity.



(a) Unprocessed trajectories

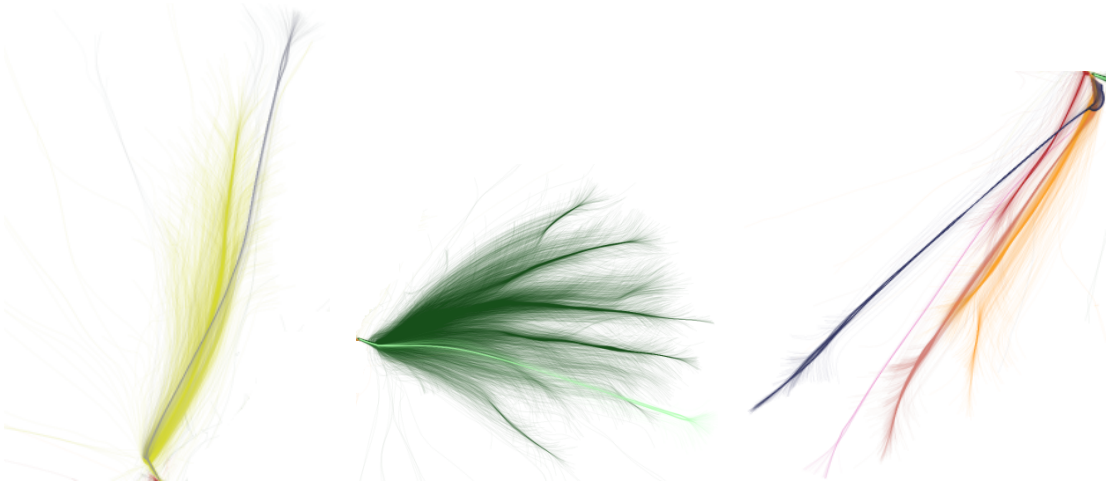
(b) Smoothed trajectories

Figure 5.19: Smoothing operation before bundling

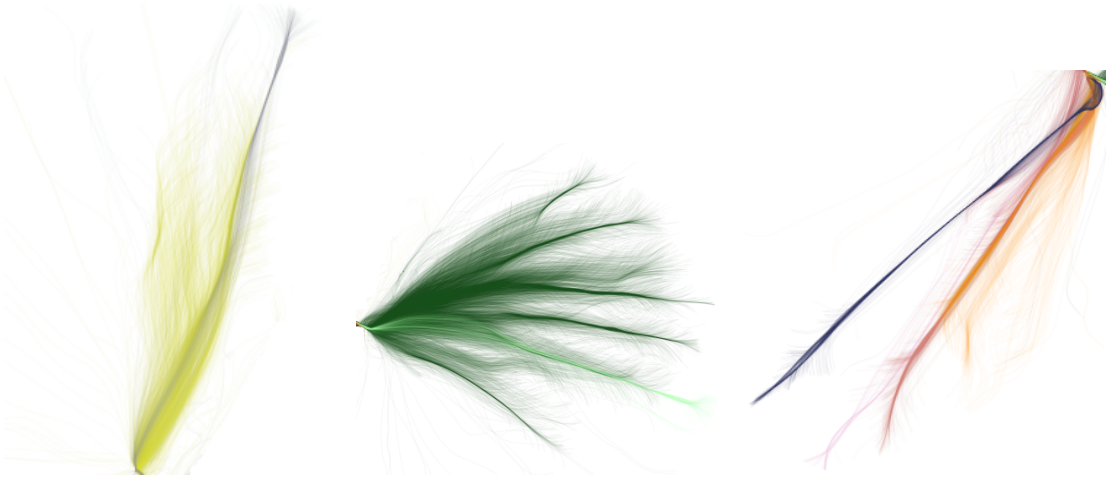


(a) Basic KDEEB **(b)** Unbundled trajectories **(c)** Improved KDEEB

Figure 5.20: Comparison Between Basic KDEEB Bundling and Improved KDEEB bundling



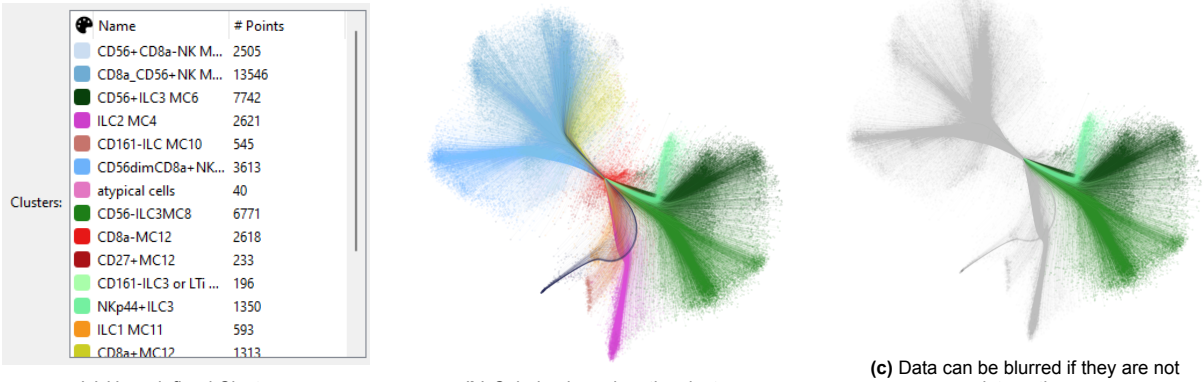
(a) Class-based bundling - example 1 **(b)** Class-based bundling - example 2 **(c)** Class-based bundling - example 3



(d) Standard bundling - example 1 **(e)** Standard bundling - example 2 **(f)** Standard bundling - example 3

Figure 5.21: Comparison between Class-based Bundling and Standard Bundling. Done with resolution 200, advection factor 3 for 5 iterations. Only relevant clusters are displayed.

5.1.4. Coloring

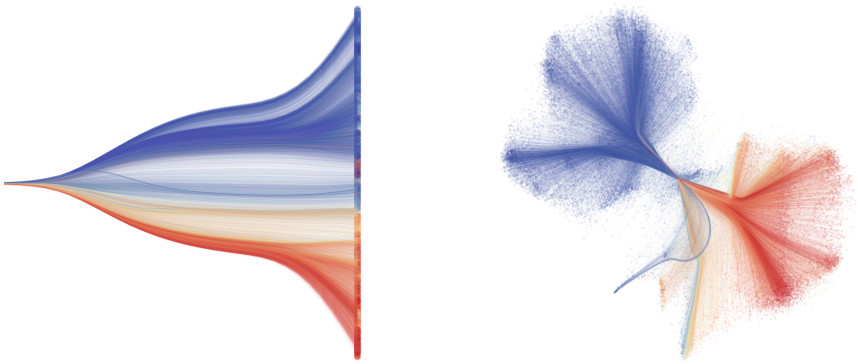


(a) User-defined Clusters

(b) Coloring based on the clusters

(c) Data can be blurred if they are not interesting

Figure 5.22: Color by Clusters.



(a) The 1D visualization colored by one of the high-dimensional feature

(b) The 2D visualization colored by one of the high-dimensional feature

Figure 5.23: Color by High-dimensional Features. Colored by value of CD127 protein marker expressions.

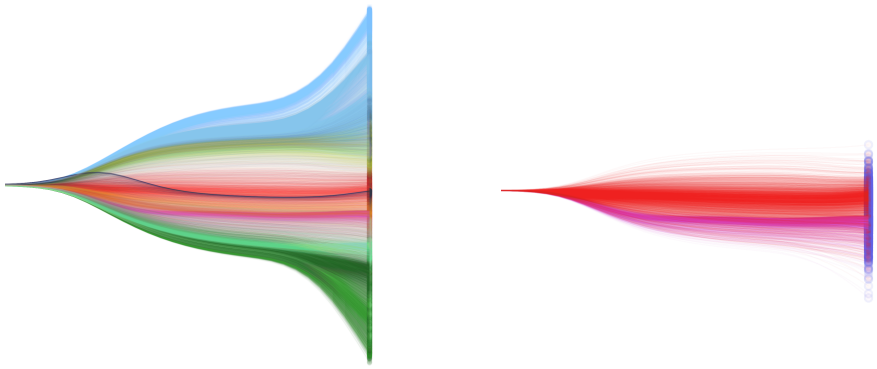
Color is used to encode information from the data, enhancing both the 2D and 1D visualizations and edge bundling. Three coloring schemes are provided. The first one is cluster-based coloring, which assigns colors based on predefined clusters. As shown in Figure 5.22a, various clusters are defined, and their assigned colors are reflected in Figure 5.22b. Additionally, users can focus on specific clusters while blurring others, as shown in Figure 5.22c. The second one is coloring based on high-dimensional features, where users can select a specific dimension from the high-dimensional input data with corresponding colors assigned via a color map. In the immune cell dataset, cells with positive marker expressions appear more reddish, while negative ones are more blueish, as shown in Figure 5.23. The third and default coloring option is constant color on all trajectories if non-of the above two options have been applied. Users can freely adjust color hues and opacities at any time and switch between these three coloring options as needed, which fulfills requirement **V4.2**.

5.1.5. Shared Interaction Components Between Visualizations

Most of the visual designs discussed so far focus primarily on static visualization rather than user interaction. In t-SNE trajectory visualization, user interaction not only reduces visual clutter [4] [47] but also enhances the relevance of the visualization for the given task, thereby better supporting all user tasks. To this end, we propose several shared components across visualizations, providing navigation, filtering, zooming, and linking features.

Navigation

In both 2D and 1D visualizations, users can navigate through the t-SNE optimization using a slider. Trajectory segments outside the selected range of iterations are omitted from the visualization and other



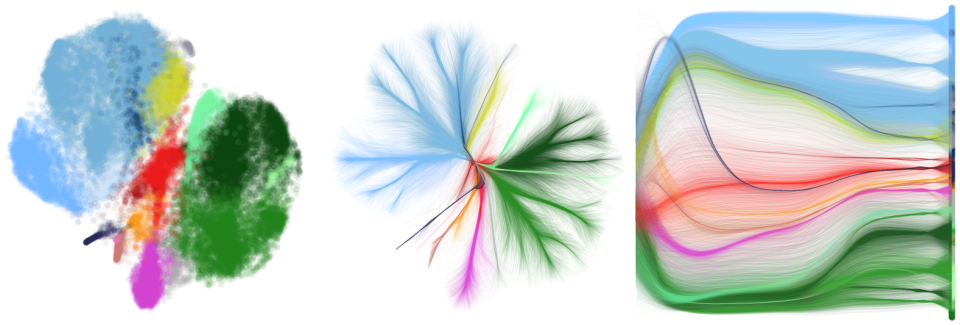
(a) A 1D visualization before filtering **(b)** A 1D visualization after filtering

Figure 5.24: Data Filtering.

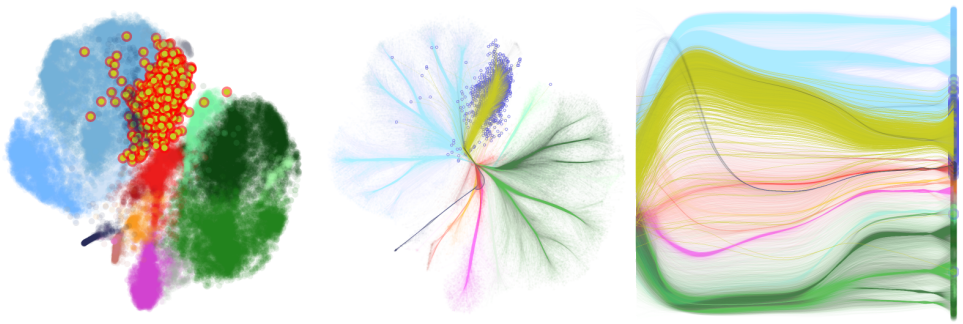
interactions. This allows users to directly access any continuous range of iterations, simplifying the visualization by focusing on the most relevant parts of the optimization process.

Filtering

Data filtering enables users to focus on particular parts of the dataset (**V4.1**). Filtering is performed through data selection, where data passing the filtering is added to a selection set and highlighted. Two filtering options are available: brushing-based filtering and cluster-based filtering. A brushing interaction allows users to select trajectories that are within or intersect the selected area. Highlighted trajectories are emphasized by increasing their opacity and adding a halo effect around the endpoints. The brushing process can be set additive, allowing users to create a complex selection set through multiple selections. Our tool also allow users to select certain classes of interest. When cluster information is available, users can select specific clusters of interest while blurring or omitting others. As shown in Figure 5.24, the right visualization highlights the red and magenta clusters, while omitting the other clusters.



(a) Visualization before creating selection



(b) Visualization with selection information shared globally

Figure 5.25: Data Linking Between Visualizations

Zooming

Zooming allows for detailed inspection of data dynamics by defining a specific area of interest (**V4.1**), improving focus on relevant details. During our experiments, we observed that trajectories from neighboring clusters often overlap, leading to visual clutter that makes it difficult to distinguish relative positions. By zooming in, users can achieve a more granular view, facilitating precise selection and inspection of trajectories.

Linking Between Visualizations

Linking ensures that public information such as selection set and colors are shared across all visualizations using the same dataset, thereby enhancing user-controlled data display (**V4**). When data is selected in one visualization, the corresponding points are highlighted in all linked visualizations. This two-way interaction between visualizations and the application framework provides users with a comprehensive understanding of the dataset and its dynamics. For instance, in a setup with multiple 1D and 2D visualizations, selecting data in one visualization will highlight the same data in other visualizations. This allows visualizations to complement each other and facilitates the comparison of different parameter settings in parallel, as shown in Figure 5.25b, where the selection of the yellow cluster is reflected across multiple visualizations.

5.2. Implementation

As mentioned earlier, our solution is developed in an iterative way. The initial version of the 2D visualization was implemented as a web application using JavaScript and D3 during the early phase of project development, as shown in Figure 5.26. However, the web application suffered from slow loading and rendering on t-SNE trajectories of human fetal intestine immune cell we use, which is considered a typical-sized t-SNE trajectory dataset. Consequently, it was not included in the final design and turned to desktop application as our solution for requirement **V3.1**. We implemented the visualization as a plugin for Manivault Studio, an extensive data analytics framework for high-dimensional data using C++, Qt, and OpenGL. The backend of the plugin, responsible for data preparation, loading, processing, and handling user interactions, is written in C++, aligning with the core system of Manivault Studio and the majority of its other plugins. The GUI is developed using Qt libraries, while the visualizations are rendered as views within corresponding GUI components using OpenGL. With this setting, handling data with order of 10^5 provides rendering in interactive time, which fulfills **V3.1**. Our data-driven edge bundling algorithm is executed in a separate OpenGL context other than the one used for view rendering. Data input or created in the splatting, density estimation, and edge advection steps is stored in textures, with the results rendered onto an offscreen buffer. t-SNE embeddings are generated through a dedicated t-SNE analysis plugin, which we modified from the existing ManiVault Studio plugin to record the full history of t-SNE embeddings across iterations. We also rewrite the binary loader of ManiVault Studio for creating clusters upon loading the input data. During experiments, we offer users the option to subsample the trajectories before running t-SNE. This improves performance, while the impact on the visualization is negligible, ensuring a balance between efficiency and visual fidelity.

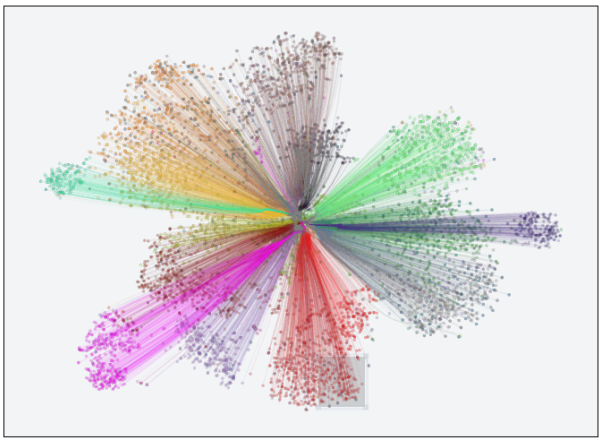


Figure 5.26: 2D visualization in early prototype with JavaScript and D3

6

Evaluation

In this section, we evaluate the effectiveness of our visualization tool through a two-part assessment. In each part, participant(s) were assigned specific tasks that required them to identify the structure or behavior of points and clusters, rather than directly generating hypotheses or discovering patterns. The first part consisted of a expert pilot study conducted by a researcher specializing in biomedical-related topics. The second part was a general study targeting users with little to no prior domain knowledge. To accommodate this, we redesigned the tasks, allowing participants to complete the tasks without requiring prior familiarity with the data. We also utilized multiple datasets in the general study. Both parts evaluate how well our visualization tool supports participants in performing data-related activities with each covering one or multiple user tasks defined in Chapter 3. The majority of the results are qualitative. In addition to collecting responses to task-related questions, we gathered open-ended feedback from the participant in the first part. In the second part, we compared users' performance and overall experience between our visualization tool and a traditional approach. To ensure consistency, we kept the parameters unchanged in the second part, allowing users to analyze identical visualizations.

6.1. Expert Pilot Study

The participant in this pilot study is an immunologist and one of the authors of Li et al.'s work. He is an expert in single-cell biology and the human fetal intestine immune cell dataset used for our evaluation. While familiar with t-SNE-like algorithms and visualizations, he was new to ManiVault Studio.

The expert pilot study aimed to gather open-ended feedback on domain-specific insights by allowing the participant to conduct tasks using our visualization tool, explore the data freely, and respond to open-ended questions in a post-test questionnaire. The questionnaire covered the participant's exploration experience, potential improvements, and general remarks. Before the test, we introduced ManiVault Studio to ensure he could complete the tasks smoothly. Following the introduction, we created a 2D view and a normalized 1D view with bundled trajectories for exploration. For each task, we provided instructions and then asked the participant to identify relevant structures and interpret them based on his prior knowledge. They are listed in Table A.1 in the appendix. A sample task was as follows:

Select the CD8a-MC12 cells (colored red) and the CD27+MC12 cells (colored dark red) in the cluster dataset. What can you infer from their behavior? How does this observation relate to your knowledge of these cell types?

During the test, the participant successfully resolved the tasks and quickly related observations from our visualizations to his immunology expertise. As task outputs, he provided several observations that aligned with established scientific knowledge, such as the early split of the CD34+MC1-2 cell group. Some findings supported his research hypotheses, while one observation appeared to offer additional insights. A detailed list of the participant's observations is provided in Table A.2 in the appendix.

At the end of the session, as well as in the questionnaire (Table A.3), the participant noted that our visualizations effectively captured the general cluster separation behavior in early phases and the emergence

of more localized variations within clusters later on. He appreciated that this behavior, observed in both our 1D and 2D views, aligned with his prior knowledge and could be valuable for studying heterogeneity and differentiation in similar datasets. He also gave positive feedback on the data filtering feature and the versatile options for focusing on interesting data. Regarding bundling techniques, he remarked that bundling is not only useful for reducing clutter but also a potential tool for revealing branching patterns, which could provide valuable biological insights. The participant concluded the test with the following remark:

"I do think that these trajectories show some kind of interesting additional relationship between these cell types that just are not inferred in the final t-SNE. And it's not only about trajectories—it's also about whether they're truly distinct or not, like that light blue one (CD56+CD8a-NK MC17), or that light green one (CD161-ILC3/LTi MC9 and NKp44+ILC3). Now I'm very curious to use this in our current project."

Overall, the feedback from this pilot study was highly positive, suggesting that our visualizations have potential for complementing traditional scatterplot animations in analyzing t-SNE dynamics.

6.2. General Study

In the general study, we evaluated our visualizations with a broader range of users to assess their effectiveness in supporting various user tasks.

6.2.1. Study Setup

For this study, we developed a tool based on the visualization study framework reVISit [12]. This framework supports customizable study layouts, multiple question types for collecting responses in different data formats, various visualization formats, and randomized question sequences to minimize memory effects between related questions.

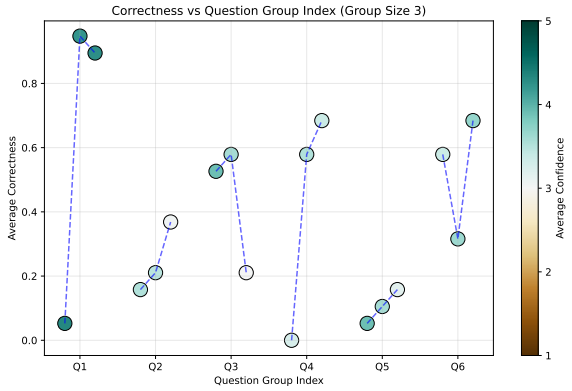
The tool was deployed as a webpage, with responses stored in a database. We analyzed only the completed questionnaires. The study was primarily distributed within TU Delft, including master's students from the Data Visualization course in the Computer Science program and members of the Computer Graphics and Visualization research group. Additional participants were current or former students from other universities, meaning most had a higher education background. To familiarize participants with the study context, we provided an introduction to dimensionality reduction, t-SNE, and t-SNE trajectories. Technical details were intentionally minimized to ensure that users without prior knowledge could still answer the questions.

6.2.2. Study Content

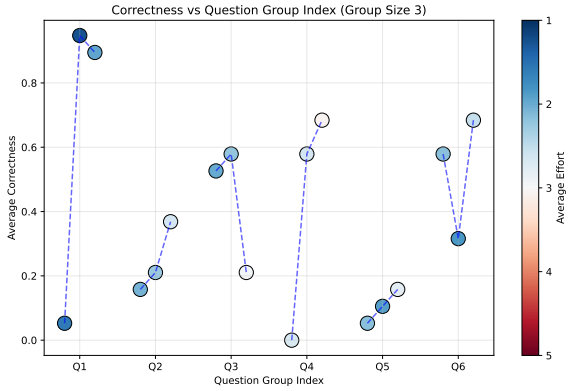
Our questions were designed to compare two key aspects of our visualization. First, we evaluated three types of visualizations: the 2D, 1D, and traditional 2D scatterplot animation. Second, we assessed the effectiveness of the visualization with and without the bundling technique. To address the first aspect, we created six questions, each presented to participants three times as a question group, once for each visualization type (the 2D, 1D, and scatterplot animation). For the second aspect, we designed four questions, each shown twice, comparing visualizations with and without bundling. This results in ten

Question	T1.1	T2.1	T2.2	T2.3
Q1			✓	
Q2	✓	✓		✓
Q3		✓		✓
Q4		✓		✓
Q5				✓
Q6				✓
Q7	✓			✓
Q8	✓			
Q9	✓			
Q10	✓			

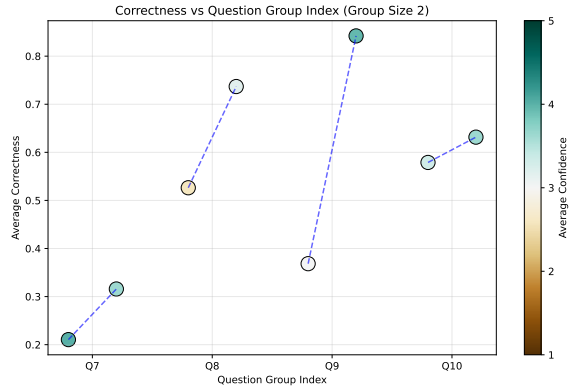
Table 6.1: Questions vs. User Tasks



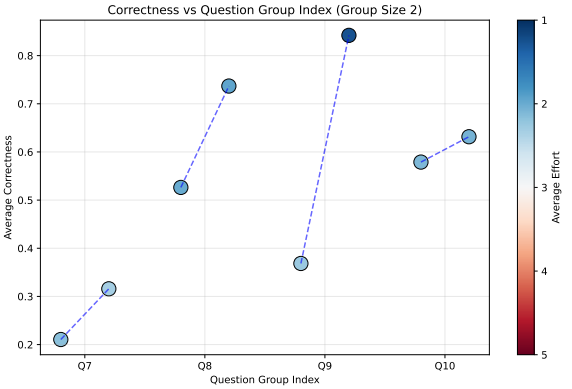
(a) correctness colored by confidence, for questions focusing on comparing 2D/1D/animation



(b) correctness colored by effort, for questions focusing on comparing 2D/1D/animation



(c) correctness colored by confidence, for questions focusing on comparing bundle/no bundle



(d) correctness colored by effort, for questions focusing on comparing bundle/no bundle

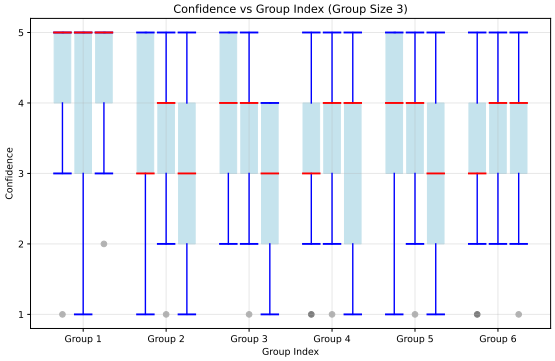
Figure 6.1: Overview on Answer Correctness. Average confidence level or effort spent is encoded as color. For groups with size 3, the three dots represent the 2D, 1D and animation, from left to right; For groups with size 2, the two dots represent the unbundled and bundled visualization, from left to right.

groups containing 26 individual questions, which are enumerated in Figure A.1 in the appendix with the mapping shown in Table A.4. Each question aimes to evaluate one or more user tasks, as shown in Table 6.1. Due to the limitations of an online test format, we did not evaluate T1.2 (Navigating the positions of embeddings over t-SNE iterations); Instead, we provided participants with pre-navigated visualizations focusing on the most relevant iterations.

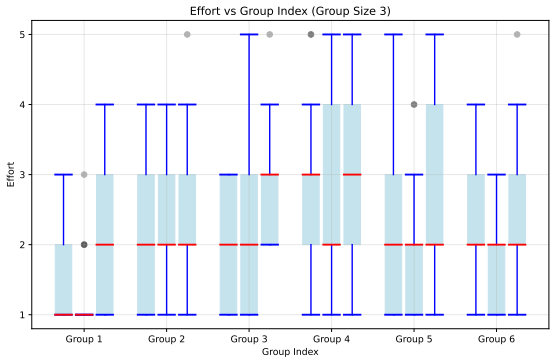
In addition to collecting responses, we recorded the effort expended for each task and the participants' confidence levels using Likert scales (ranging from 1 to 5). To further minimize memory effects, we applied varying color schemes across visualizations in different questions. During the analysis, we assumed that each response was made independently by the participants.

6.2.3. Results

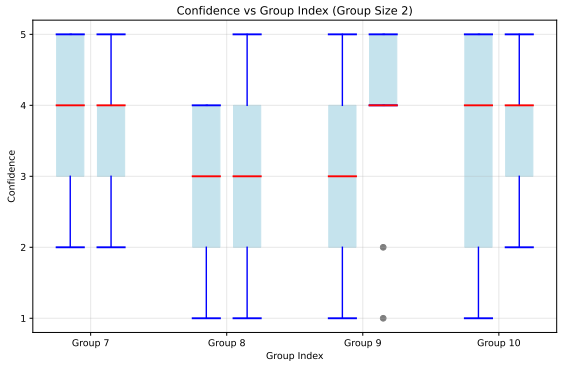
We collected 19 fully answered questionnaires. Figure 6.1 shows the correctness of user answers, while Figure 6.2 shows an overview of user confidence and effort. To assess the significance of our results in terms of correctness, confidence, and effort, we conducted statistical hypothesis testing. To compare the rate of correctness within each question group, we ran Fisher's exact tests. The null hypothesis (H_0^F) states that the proportion of correct answers is the same for both visualizations in the question, while the alternative hypothesis (H_A^F) posits that the proportion of correct answers differs bet-



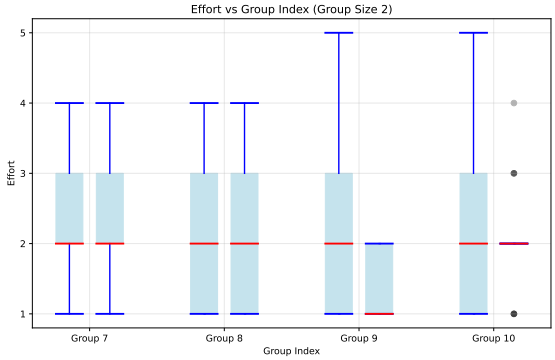
(a) confidence level for questions focusing on comparing 2D/1D/animation (18 questions)



(b) effort spent for questions focusing on comparing 2D/1D/animation (18 questions)



(c) confidence level for questions focusing on comparing bundle/no bundle (8 questions)



(d) effort spent for questions focusing on comparing bundle/no bundle (8 questions)

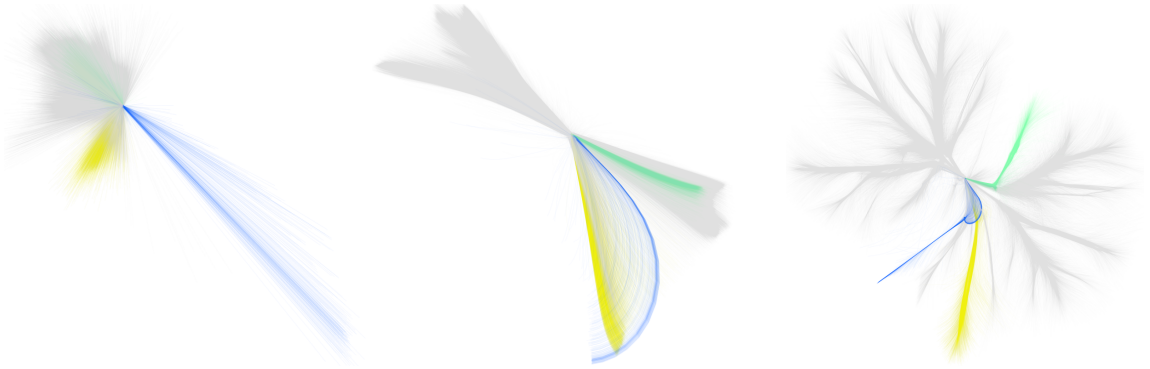
Figure 6.2: Overview on Answer Confidence and Effort. We display and compare the average confidence level and effort spent within each group of questions.

ween the two visualizations in the question. The null hypothesis (H_0^t) states that the mean confidence level (or effort) remains the same for both visualizations in the question, whereas the alternative hypothesis (H_A^t) posits that the mean confidence level (or effort) differs between visualizations in the question. A threshold p-value of 0.05 was applied, and all tests followed a two-tailed approach.

Comparing the 2D, 1D, and Animation Visualizations

The top two illustrations in Figure 6.1 reveal that animation achieves the highest correctness in 4 out of 6 question groups (groups 2, 4, 5, and 6). In the remaining two groups (1 and 3), the 1D visualization demonstrates the highest correctness. While the 2D visualization exhibits comparable correctness in some cases, it generally underperforms compared to the other two methods. Users also generally expended less effort when answering questions with the 2D or 1D visualizations than with animation. However, no clear pattern emerges regarding confidence, as shown in the top left illustration.

Fisher's exact test indicates significantly higher correctness for both the 1D visualization and the animation compared to the 2D visualization in question group 1, where participants had to identify the order of splits (T2.2). We hypothesize that the movement of 2D data points in varying directions and speeds during optimization makes it difficult for users to track data positions at a given iteration, confirming the limitations of the 2D visualizations discussed in previous chapters. Specifically, in the 2D visualization of question group 1, as illustrated in Figure 6.3, users might mistakenly identify the green cluster as the first to split when provided with the right-most illustration, because the trajectories of the blue and yellow clusters nearly overlap near the origin. However, the blue cluster moves significantly faster than the yellow and is actually the first to split, as shown in the left most illustration. Later, the yellow cluster passes through the previous positions of the blue cluster, creating the illusion that they moved



(a) Phase 1 **(b)** Phase 2 **(c)** Phase 3

Figure 6.3: Limitation of the 2D Visualization in Question Group 1. Iteration increases from the left to the right.

together initially. This suggests that the 2D visualization is less effective in identifying early split behavior and other iteration-related patterns. Additionally, although the 1D correctness was only slightly higher than animation in this case, users expended significantly less effort with 1D than animation, potentially supporting the preference for 1D visualizations in similar tasks.

Similarly, in question group 4, both the 1D visualization and the animation achieved significantly higher correctness than the 2D visualization. Since this task also involved recognizing iteration-related behavior (as in question group 1), most participants struggled to pinpoint the exact iteration phase in which the split occurred (T2.1). Specifically, in the 2D visualization, users might mistakenly perceive one cluster as more closely related to the target cluster due to their overlapping trajectories. However, in reality, the cluster separates earlier, while another cluster and the target cluster remain closer throughout the iterations. This challenge was further compounded by trajectory bundling, which distorted spatial relationships and made it harder to assess proximity between clusters (T2.3). During the study, participants were provided only with line-strip representations of trajectories. To mitigate misconceptions about spatial relationships and improve proximity judgments, we propose enhancing the 2D visualization by overlaying point markers (e.g., circles) to provide clearer positional references, similar to traditional scatterplots.

In question group 3, the 1D visualization demonstrated significantly higher correctness than the animation. Tasks in this group required recognizing branching structures, identifying shared dynamics between clusters (T2.3), and identifying splitting behavior (T2.1). Additionally, both the 1D and 2D visualizations required significantly less effort than animation. We speculate that the explicit trajectory representation in both the 1D and 2D, along with the iteration encoding in the 1D, made it easier to identify dynamic behavior, particularly when multiple clusters were involved. In contrast, the scatterplot animation was less effective at conveying branching behavior, as users had to mentally reconstruct trajectories based solely on moving scatter points—an especially challenging task when dealing with multiple clusters. As a result, the scatterplot animation imposed a higher cognitive load, making the task more demanding.

One interesting but not statistically significant finding is that the 2D visualization outperforms the 1D visualization in question group 6—the only instance among all six question groups. In this question, we presented two clusters that were separate but highly related. The 2D visualization effectively conveyed their proximity: they were more closely related than other clusters but still distinct. However, because the 1D visualization represents the embedding using a single dimension, such clusters may appear less separable due to overlaps caused by the limited space, leading to misinterpretations.

Comparing the Not Bundled and Bundled Visualizations

For these questions, users were given unbundled and bundled trajectory visualizations of closely related clusters to assess their effectiveness in identifying branches. The bottom two illustrations of Figure 6.1 indicate that bundled visualizations led to higher correctness across all four groups of comparisons. Confidence was generally higher with bundled visualizations, except in question group 7. Effort was lower for bundled visualizations in the same groups with higher confidence, though the difference was less pronounced than for confidence.

Although we observed increasing correctness in all 4 question groups after bundling, the only significant difference in correctness occurred in question group 9. In the unbundled visualization, participants struggled with ambiguous cluster structures, resulting in a low correct response rate (36.8%), with 31.6% unable to answer. In contrast, with bundled trajectories, only 5% of participants failed to respond, while 84.2% answered correctly. Additionally, in this question group, confidence was significantly higher and effort lower for the bundled visualization compared to the unbundled. We believe that the bundling technique effectively reduced visual clutter, making summarized structures easier to interpret. This, in turn, led to improvements in correctness, confidence, and effort reduction. These results support the notion that edge bundling enhances users' ability to discern branching structures in t-SNE development.

6.2.4. Discussion

Reflecting on the tasks defined in Chapter 3, we conclude that the 2D visualization enables users to analyze relationships between clusters (T2.3) when iteration information is either unnecessary or not misleading. However, when iteration details are required, its effectiveness diminishes compared to the 1D or animation. While the 2D visualization helps identify *which* cluster splits (T2.1), it does not effectively convey *when* the split occurs. Also, users struggled to interpret trajectory similarity and proximity in the 2D visualization (T2.3). In contrast, the 1D visualization is advantageous for a broader range of tasks, including determining which clusters split and when (T2.1), establishing the order of splits (T2.2), and analyzing relationships between clusters (T2.3). Edge bundling improves data summarization (T1.1) by reducing visual clutter and clearer structures.

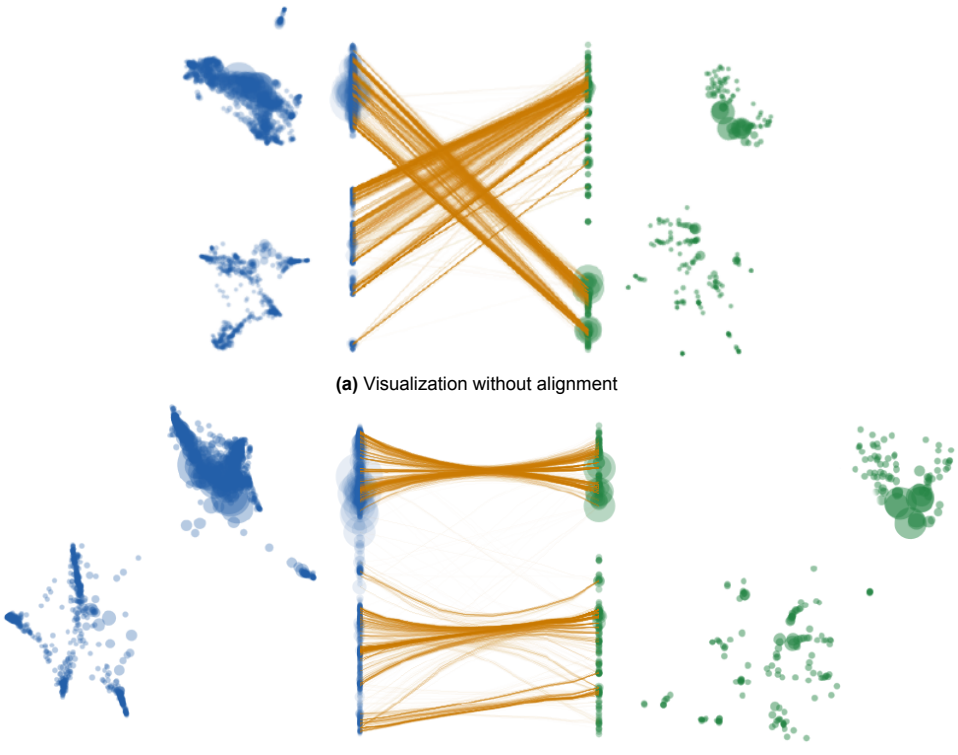
Overall, our study reveals that while individual visualizations sometimes yield comparable results, animation generally outperforms both the 2D and 1D representations. This is because the 2D and 1D visualizations are more specialized and task-dependent, whereas animation covers a broader range of tasks. Additionally, edge bundling improves correctness and may enhance task efficiency and confidence.

One limitation of our general study is that participants were restricted to a single static view at a time. For the same reason, task T1.2 (Navigating the positions of the embeddings over t-SNE iterations) was not evaluated. Allowing users to switch between the 1D and 2D visualizations or interact with them dynamically may yield different insights and should be explored in future work. Additionally, the animation was provided as a GIF without controls for speed adjustment, rewinding, or pausing. This limitation may have impacted participants' ability to analyze dynamic changes effectively. Finally, our study involved 19 participants, which may limit the statistical power of our results. While significant trends were observed, a larger sample size would provide stronger statistical validation and generalizability of findings. Future studies should aim for increased participation to bolster the robustness of conclusions.

7

Reflection and Conclusion

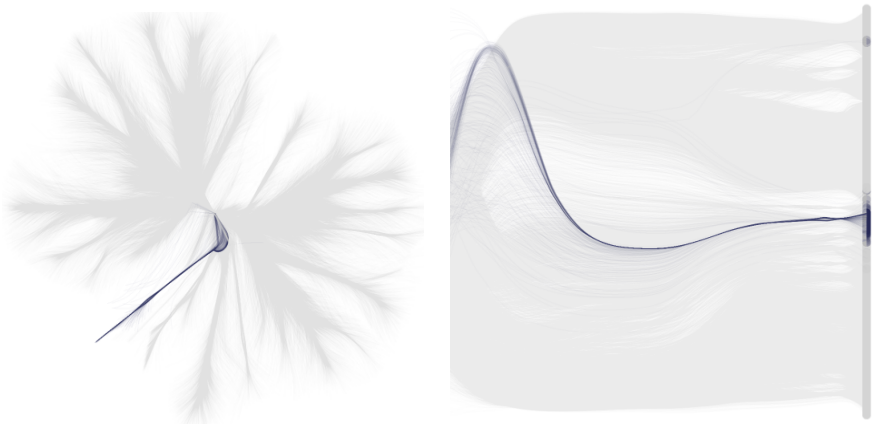
Although our visualizations received positive feedback from domain expert and demonstrated promising results in the general study, they also have certain limitations. First, as discussed in Chapter 5, trajectories in the 2D and the 1D visualizations may not align well due to different initializations. Aligning the two visualizations could be an interesting direction for future work. One possible approach is to introduce constraints in the t-SNE optimization process to prevent corresponding points from drifting too far apart. Pezzotti et al. [36] modified the t-SNE cost function by adding a squared distance term between the positions of 1D points and the vertical positions of 2D points, thereby penalizing large discrepancies between 1D and 2D embeddings. As shown in Figure 7.1, such technique successfully aligns four embeddings (two 2D and two 1D). A similar strategy could be applied in our case to align a single 2D embedding with a single 1D embedding.



(a) Visualization without alignment

(b) Visualization with alignment

Figure 7.1: Aligning multiple embeddings



(a) 2D visualization **(b)** 1D visualization

Figure 7.2: Information loss due to lower detail level of the 1D visualization

Second, due to the loss of detail in lower-dimensional t-SNE embeddings, 1D trajectories are sometimes less accurate in representing data dynamics compared to the 2D trajectories. For example, in Figure 7.2, the dark navy cluster (the CD34 cell group) separates from the others and occupies a distinct space in the 2D visualization. However, in the 1D visualization, it undergoes significant vertical displacement, crossing multiple other clusters before stabilizing. This limitation arises from the restricted space in 1D. Therefore, it is recommended to refer to the 2D visualization when interpreting 1D trajectories.

Third, our implementation of the bundling algorithm relies on a basic GPU rendering pipeline, which limits its performance. While it is suitable for most devices, further optimization, such as CUDA acceleration [45], could significantly enhance efficiency on supported hardware, reducing bundling computation times from seconds to milliseconds. Additionally, there is room for improvements in the data-driven bundling technique, as we have identified artifacts in the bundled trajectories, mentioned in Chapter 5. Future work could explore more effective bundling mechanisms that minimize such artifacts while balancing clutter reduction and information loss.

Furthermore, feedback from our pilot user highlighted the need for more explicit guidance in interpreting (bundled) trajectory visualizations. Specifically, users would benefit from understanding which attributes in the high-dimensional data drive certain structural patterns, such as splitting or branching. We have implemented a coloring option based on high-dimensional attributes. Future work could explore integrating this information more seamlessly into cluster coloring, allowing users to view both cluster information and relevant high-dimensional attributes within a single visualization.

Finally, our visualizations were tested on a small scale. More comprehensive testing is needed to evaluate the overall performance across diverse datasets and use cases.

In conclusion, we designed and implemented the 2D and 1D visualizations, along with supporting components, to facilitate visual analytics of t-SNE dynamics, making it easier to identify meaningful structures. To enhance clarity, we improved an existing bundling algorithm to preserve key structures while reducing visual clutter. Our design was guided by a analysis of user tasks and requirements. To evaluate our approach, we conducted both pilot and general user studies, receiving positive feedback and demonstrating its potential to simplify task execution compared to traditional workflows. Additionally, we have discussed the limitations of our work and proposed potential directions for future improvements.

8

Acknowledgments

I thank T. Höllt, my mentor, for providing guidance throughout this thesis project; V. van Unen from Leiden University Medical Center for participating in the expert pilot study as part of our evaluation.

Additionally, I acknowledge the use of ChatGPT for assisting in refining and improving the clarity of the text in this thesis.

References

[1] Natalia Adrienko and Gennady Adrienko. “Spatial generalization and aggregation of massive movement data”. In: *IEEE Transactions on visualization and computer graphics* 17.2 (2010), pp. 205–219.

[2] Natalia Andrienko, Gennady Andrienko, and Salvatore Rinzivillo. “Exploiting spatial abstraction in predictive analytics of vehicle traffic”. In: *ISPRS International Journal of Geo-Information* 4.2 (2015), pp. 591–606.

[3] Kaye Enid Basford and John Wilder Tukey. *Graphical analysis of multi-response data*. Chapman and Hall/CRC, 2020.

[4] Richard A. Becker, Stephen G. Eick, and Allan R. Wilks. “Visualizing network data”. In: *IEEE Transactions on visualization and computer graphics* 1.1 (1995), pp. 16–28.

[5] Ilya Boyandin, Enrico Bertini, and Denis Lalanne. “Using flow maps to explore migrations over time”. In: *Geospatial visual analytics workshop in conjunction with the 13th AGILE international conference on geographic information science*. Vol. 2. 3. 2010.

[6] Matthew C Cieslak et al. “t-Distributed Stochastic Neighbor Embedding (t-SNE): A tool for eco-physiological transcriptomic analysis”. In: *Marine genomics* 51 (2020), p. 100723.

[7] William S Cleveland. *Visualizing data*. Hobart press, 1993.

[8] Dorin Comaniciu and Peter Meer. “Mean shift: A robust approach toward feature space analysis”. In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619.

[9] Kristin A Cook and James J Thomas. *Illuminating the path: The research and development agenda for visual analytics*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005.

[10] Weiwei Cui et al. “Geometry-based edge clustering for graph visualization”. In: *IEEE transactions on visualization and computer graphics* 14.6 (2008), pp. 1277–1284.

[11] George Dimitriadis, Joana P Neto, and Adam R Kampff. “t-SNE visualization of large-scale neural recordings”. In: *Neural computation* 30.7 (2018), pp. 1750–1774.

[12] Yiren Ding et al. “reVISit: Supporting Scalable Evaluation of Interactive Visualizations”. In: *2023 IEEE Visualization and Visual Analytics (VIS)*. IEEE. 2023, pp. 31–35.

[13] David L Donoho et al. “High-dimensional data analysis: The curses and blessings of dimensionality”. In: *AMS math challenges lecture* 1.2000 (2000), p. 32.

[14] Ozan Ersoy et al. “Skeleton-based edge bundling for graph visualization”. In: *IEEE transactions on visualization and computer graphics* 17.12 (2011), pp. 2364–2373.

[15] Samiha Fadloun et al. “TrajectoryVis: A visual approach to explore movement trajectories”. In: *Social Network Analysis and Mining* 12.1 (2022), p. 53.

[16] Elif E Firat et al. “P-lite: A study of parallel coordinate plot literacy”. In: *Visual Informatics* 6.3 (2022), pp. 81–99.

[17] Emden R Gansner et al. “Multilevel agglomerative edge bundling for visualizing large graphs”. In: *2011 IEEE Pacific Visualization Symposium*. IEEE. 2011, pp. 187–194.

[18] Zhao Geng et al. “Angular histograms: Frequency-based visualizations for large, high dimensional data”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2572–2580.

[19] Diansheng Guo. “Flow mapping and multivariate visualization of large spatial interaction data”. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1041–1048.

[20] David Hilbert. “Über die stetige Abbildung einer Linie auf ein Flächenstück”. In: *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte* (1935), pp. 1–2.

[21] Danny Holten and Jarke J Van Wijk. “Force-directed edge bundling for graph visualization”. In: *Computer graphics forum*. Vol. 28. 3. Wiley Online Library. 2009, pp. 983–990.

[22] Harold Hotelling. “Analysis of a complex of statistical variables into principal components.” In: *Journal of educational psychology* 24.6 (1933), p. 417.

[23] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. “Graph bundling by kernel density estimation”. In: *Computer graphics forum*. Vol. 31. 3pt1. Wiley Online Library. 2012, pp. 865–874.

[24] Alfred Inselberg. “The plane with parallel coordinates”. In: *The visual computer* 1 (1985), pp. 69–91.

[25] Daniel Keim et al. *Visual analytics: Definition, process, and challenges*. Springer, 2008.

[26] Antoine Lambert, Romain Bourqui, and David Auber. “Winding roads: Routing edges into bundles”. In: *Computer graphics forum*. Vol. 29. 3. Wiley Online Library. 2010, pp. 853–862.

[27] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[28] Antoine Lhuillier, Christophe Hurter, and Alexandru Telea. “State of the art in edge and trail bundling techniques”. In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 619–645.

[29] Na Li et al. “Mass cytometry reveals innate lymphoid cell differentiation pathways in the human fetal intestine”. In: *Journal of Experimental Medicine* 215.5 (2018), pp. 1383–1396.

[30] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[31] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[32] C.-J. Minard. *Tableaux graphiques et cartes figuratives. Bibliothèque numérique patrimoniale des ponts et chaussées*. 1864.

[33] Eliakim Hastings Moore. “On certain crinkly curves”. In: *Transactions of the American Mathematical Society* 1.1 (1900), pp. 72–90.

[34] Guy M Morton. “A computer oriented geodetic data base and a new technique in file sequencing”. In: (1966).

[35] Giuseppe Peano and G Peano. *Sur une courbe, qui remplit toute une aire plane*. Springer, 1990.

[36] Nicola Pezzotti et al. “Multiscale visualization and exploration of large bipartite graphs”. In: *Computer Graphics Forum*. Vol. 37. 3. Wiley Online Library. 2018, pp. 549–560.

[37] Doantam Phan et al. “Flow map layout”. In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE. 2005, pp. 219–224.

[38] Paulo E Rauber et al. “Visualizing the hidden activity of artificial neural networks”. In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), pp. 101–110.

[39] *Space Filling Curves*. `https://en.wikipedia.org/wiki/Space-filling_curve`.

[40] Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[41] Alex Telea and Michael Behrisch. “Visual Exploration of Large Multidimensional Trajectory Data”. English. In: *Data Science for Migration and Mobility Studies*. Oxford University Press, 2022, pp. 241–266. DOI: `10.5871/bacad/9780197267103.003.0011`.

[42] Alexandru Telea and Ozan Ersoy. “Image-based edge bundles: Simplified visualization of large graphs”. In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library. 2010, pp. 843–852.

[43] Waldo R Tobler. “Experiments in migration mapping by computer”. In: *The American Cartographer* 14.2 (1987), pp. 155–163.

[44] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[45] Matthew Van Der Zwan, Valeriu Codreanu, and Alexandru Telea. “CUBu: Universal real-time bundling for large graphs”. In: *IEEE transactions on visualization and computer graphics* 22.12 (2016), pp. 2550–2563.

[46] “Visualizing multidimensional data with glyph SPLOMs”. In: *Computer Graphics Forum*. Vol. 33. 3. Wiley Online Library. 2014, pp. 301–310.

[47] Nelson Wong, Sheelagh Carpendale, and Saul Greenberg. “Edgelens: An interactive method for managing edge congestion in graphs”. In: *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714)*. IEEE. 2003, pp. 51–58.

[48] Bo Zhou and Wenfei Jin. “Visualization of single cell RNA-seq data using t-SNE in R”. In: *Stem Cell Transcriptional Networks: Methods and Protocols* (2020), pp. 159–167.

A

Evaluation Details

A.1. Expert Pilot Study

Table A.1: Tasks in the Expert Pilot Study

Tasks
Select the *CD8a- MC12* cells (in color red) and the *CD27+ MC12* cells (in color dark red) in the cluster dataset. What can you interpret from their behavior? How does this observation relate to your knowledge in these cell types?
Select the *CD34+ MC1-2* cells (in color navy blue) in the cluster dataset. What can you interpret from the behavior of this cluster? How does this observation relate to your knowledge in this cell type?
Reset the current selection and examine globally on the visualization. Looking at the *NK* (in various shades of blue) cell clusters as opposed to *CD127+ ILC* (ILC1, ILC2, ILC3) clusters, what observation do you have on their behavior? Do the two groups of clusters share part of their trajectories or they separate early? How does this observation relate to your knowledge in these cell types?
Set *Plot settings → Line opacity unselected* to 0. What can you observe inside the cluster *CD56+ CD8a- NK MC17* (sky blue) in terms of branching/splitting and how do you interpret it with your knowledge in the cell type? You may set *Plot settings → Line opacity unselected* back to 0.01.
Individual cluster splits from a larger group of multiple clusters during the optimization. Rank the order of occurrence of such splitting behavior for the following clusters of cells: *CD56+ CD8a- NK MC17* (light blue), *ILC2 MC4* (magenta), *CD34+ MC1-2* (navy blue), *CD8a+ MC12* (yellow), *NKp44+ ILC3* (light green), and *CD56 dim CD8a+ NK MC16* (mid blue).
Looking at the *NK* (various shades of blue), *ILC1 MC11* (orange), *ILC2 MC4* (magenta), and *ILC3* (various shades of green) clusters, can you interpret how they gradually show the structure of them with the *Int-ILC* cells (red, yellow)? How does this relate to your knowledge in these cell types?
Set *Plot settings → Line opacity unselected* to 0. From the behavior of the *ILC2 MC4* cells (in color magenta), the majority of *ILC3* cells (in various shades of green) and the *CD45RA+ ILC3 MC7* cells (in color grayish green), how do you interpret the relationships of these cell clusters? How does this relate to your knowledge in these cell types? You may set *Plot settings → Line opacity unselected* back to 0.01.
Look at the *CD8a- int-ILC* (in color red), the *CD8a+ int-ILC* (in color yellow) and the *NK* cells, do they share part of their trajectories? How do you interpret the relationships of these cell clusters? How does this relate to your knowledge in these cell types?

Table A.2: Participant Observations in the Expert Pilot Study

Participant Observations
The participant quickly identified the early split of the *CD34+ MC1-2* cell group. Additionally, based on their immunology expertise, the participant suggested a potential relationship between the *CD34+ MC1-2* and *CD8a- MC12* cell groups. This insight is supported by their close alignment in the 1D visualization, whereas in the 2D or scatterplot visualizations, the relationship was less apparent.
The *CD161- ILC3 or LTi MC9* and *NKp44+ ILC3* cell groups may exhibit greater heterogeneity than previously assumed. The participant observed that these cells separated into two distinct groups, as indicated by the differing trajectory directions in the 2D visualization. This observation was further reinforced by a similar splitting pattern in the 1D visualization.
The participant observed that the *CD56+ CD8a- NK MC17* cell group splits into two sub-clusters, one of which shares trajectories with the *CD56 dim CD8a+ NK MC16* group. He believed this finding offers additional insights into the order of cell differentiation.
The participant observed that the split between *ILC1 MC11* and *CD8a- MC12* occurred at an early phase in the 1D view. However, this split was less apparent in the 2D view due to the limited display area in the early t-SNE phases.
The trajectories of the *ILC2 MC4* and *CD8a- MC12* cell groups in the 2D view suggest the possibility that *CD8a- MC12* cells differentiate into *ILC2 MC4*, aligning with a hypothesis in the participant's current research. This hypothesis is further supported by the positioning of *ILC2 MC4* cells between *CD8a- MC12* and *ILC3* cells in the 1D view, considering that *CD8a- MC12* cells are known to differentiate into *ILC3* cells.
In the 2D view, the participant observed that the *CD8a- MC12* group connects to the *NK* group through the *CD8a+ MC12* group, aligning with established facts and hypotheses in their study. They also noted that the neighboring relationship between the *CD8a- MC12* and *CD8a+ MC12* groups is more clearly displayed in the 2D view than in the 1D view.
The participant described the order of splits among the specified cell groups: *CD34+ MC1-2* split first, followed by the *NK* and *ILC3* cells. The *CD8a+ MC12* group followed the movement of *NK* cells, while the *ILC2 MC4* group followed the movement of *ILC3* cells.

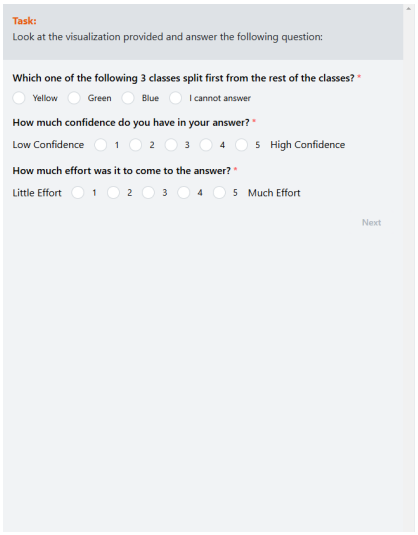
Table A.3: Feedback Received from the Expert Pilot Study

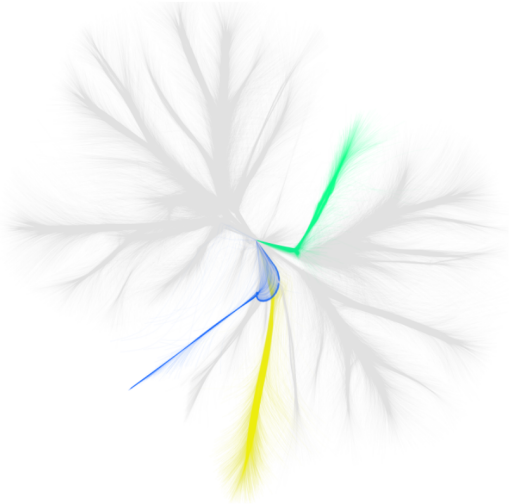
Questions	Participant Answers
What do you think of the bundling technique in improving cluster visibility and helping identify cluster dynamics in the trajectory data?	The bundling technique was very helpful in visualizing branching patterns of the trajectories. It was useful in obtaining potential novel biological insight.
What additional information do you find the 1D view provides on cluster development compared to the 2D view?	The 1D view helped identifying which cell clusters would first spread out, suggesting highly distinct patterns compared to other cell clusters.
Do the 1D and 2D views simplify data exploration and (or) reduce cognitive load as opposed to the scatterplot visualization?	The 1D and 2D views are quite related but show different subtleties, while both these views are complementary to the final t-SNE scatter plot where trajectories are not prioritized.
Is there any additional information you think that our trajectory visualizations should carry? Are there any existing features that could be improved or additional features that can be added to help you perform the tasks?	Guidance into which protein expression profiles dictate the formation and branching points of the trajectories would be helpful. The single protein expression overlays already provide some clues into this direction.

Continued on next page

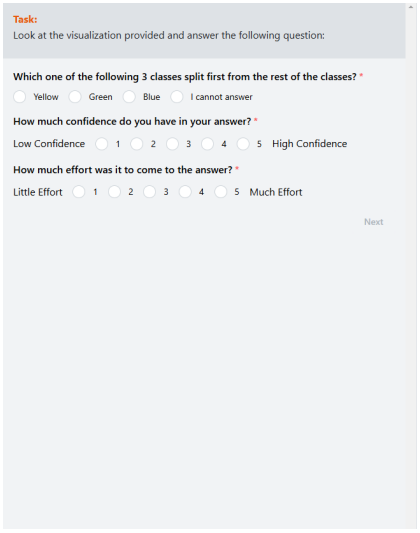
Questions	Participant Answers
Did the visualization provide any additional insights or structures on the cell data that were not mentioned in the tasks asked?	Yes. Additional insights were gained from a cell cluster we thought was homogeneous before, which seemed to contain two subsets based on displaying two different trajectory patterns. This helped identify that this particular cell cluster may contain two cell types within and thereby offer guidance into refining the cell clustering strategy.
Could you provide general feedback or comments in solving the tasks in your work based on the overall experiences of using this visualization tool?	I was very impressed by this software and would love to implement it. The data used was based on CyTOF data. However, we now transitioned to spectral-flow-cytometry data to obtain similar types of data (single cell data points assessed for 40 protein expression) but through a different technique. I would be curious to see if the software is versatile in analyzing trajectories within spectral flow cytometry data in a similar fashion.

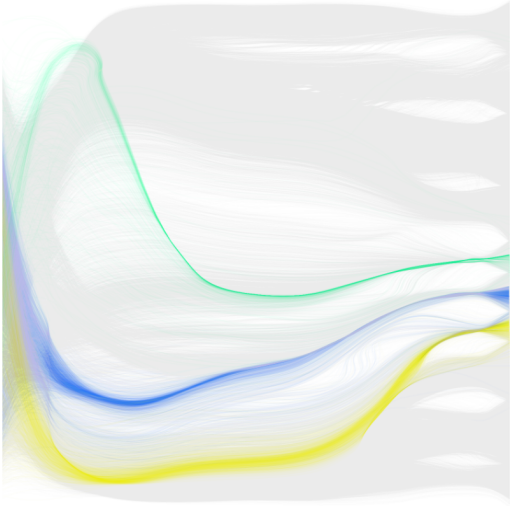
A.2. General Study



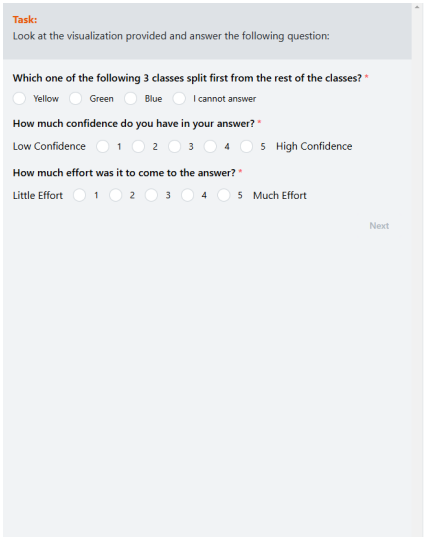


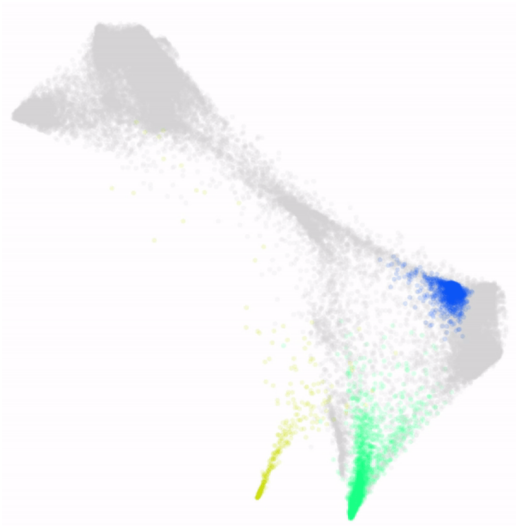
(a) Question 1



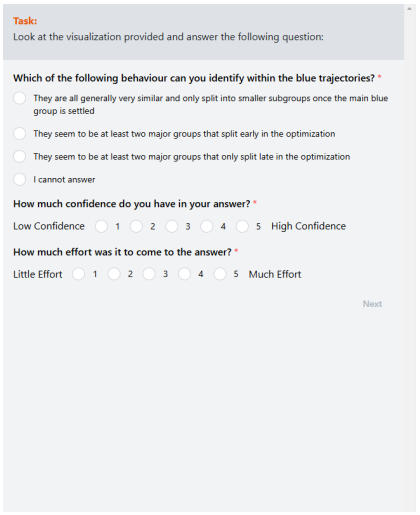


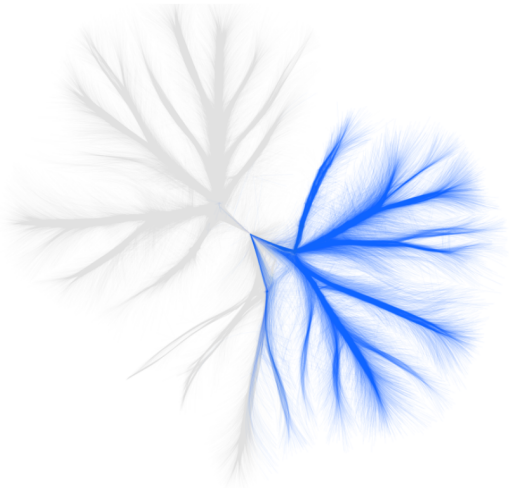
(b) Question 2



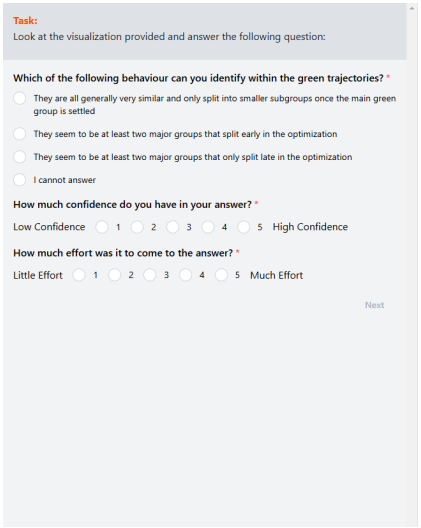


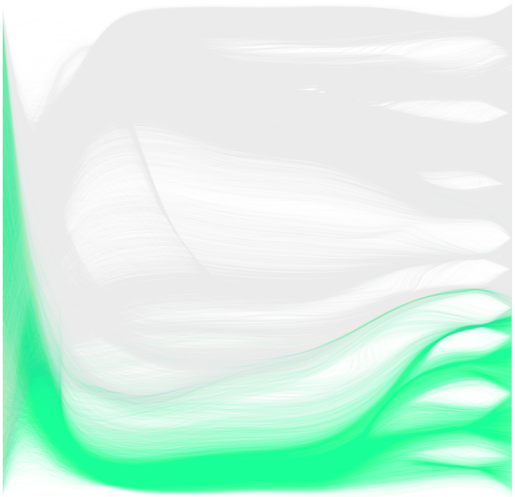
(c) Question 3



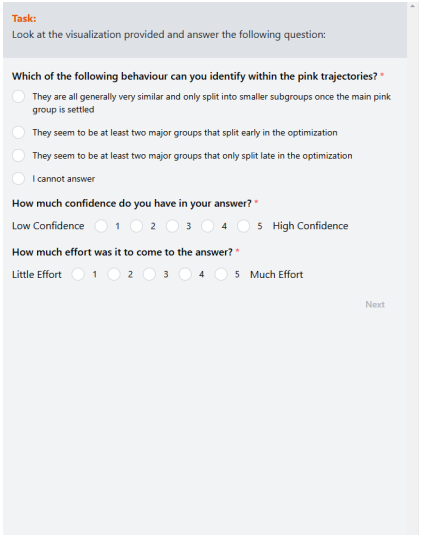


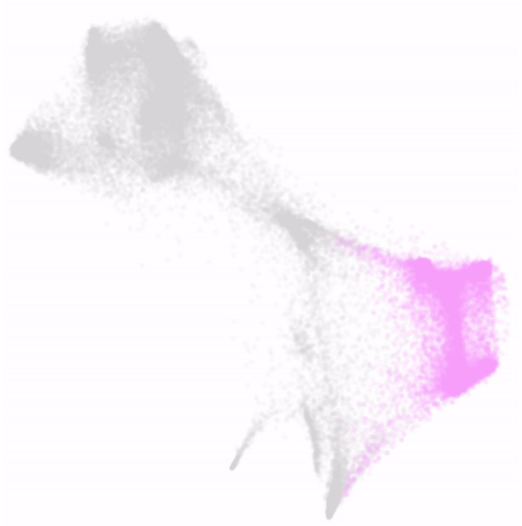
(d) Question 4



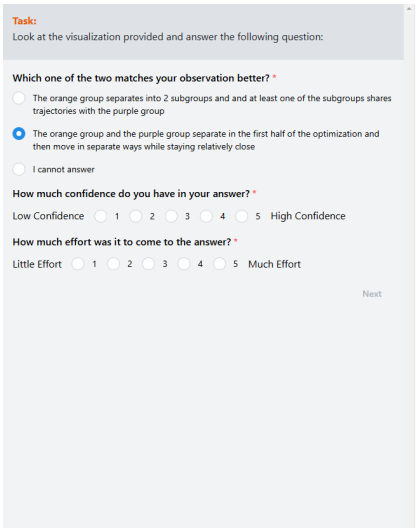


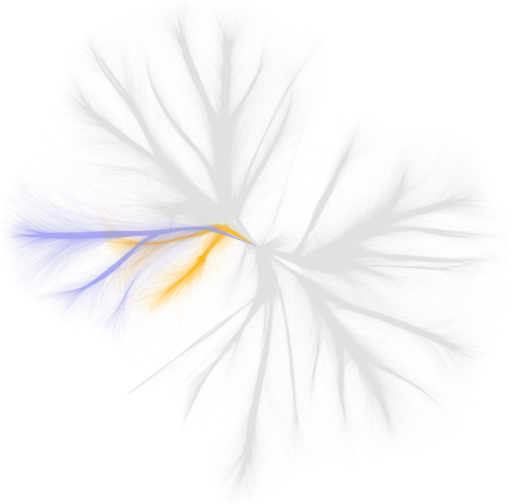
(e) Question 5



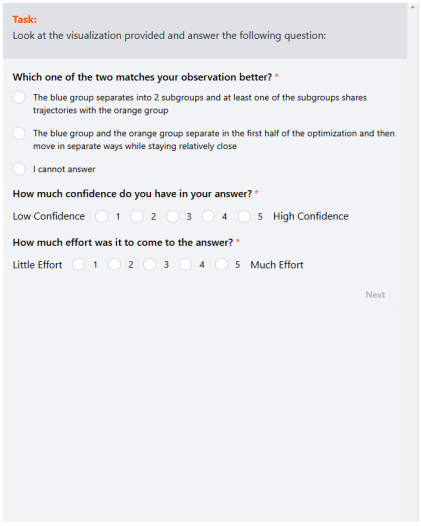


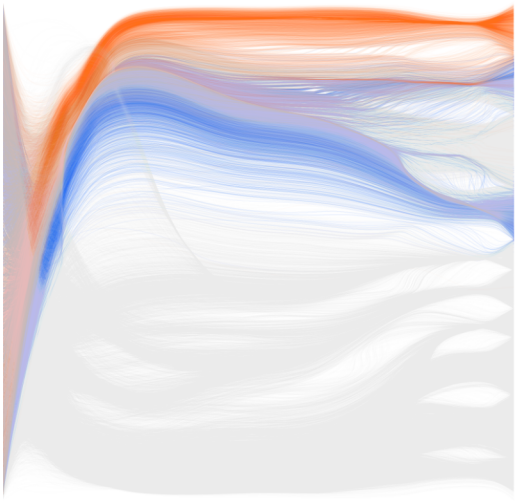
(f) Question 6



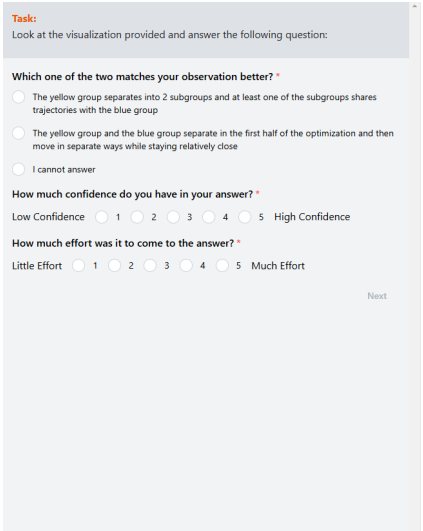


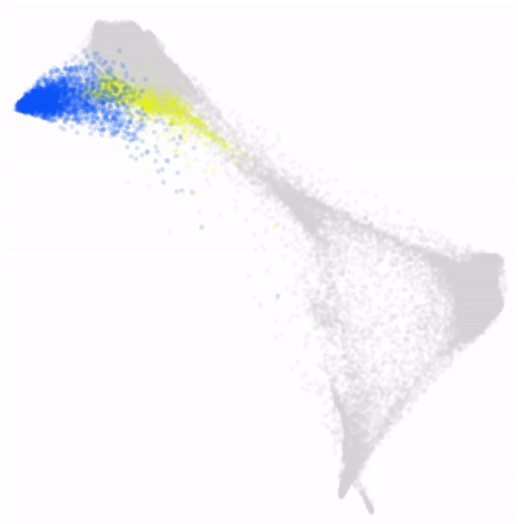
(g) Question 7





(h) Question 8





(i) Question 9

Task:
Look at the visualization provided and answer the following question:

Which of the following behaviour can you identify among the three groups, assuming there are 2 phases (earlier, later) that equally divide the optimization? *

- They share trajectories before they split at the earlier phase and in the end, the majority of the orange group is more related to the blue group than the red group
- They share trajectories before they split at the later phase and in the end, the majority of the orange group is more related to the blue group than the red group
- They share trajectories before they split at the earlier phase and in the end, the majority of the orange group is more related to the red group than the blue group
- They share trajectories before they split at the later phase and in the end, the majority of the orange group is more related to the red group than the blue group
- I cannot answer

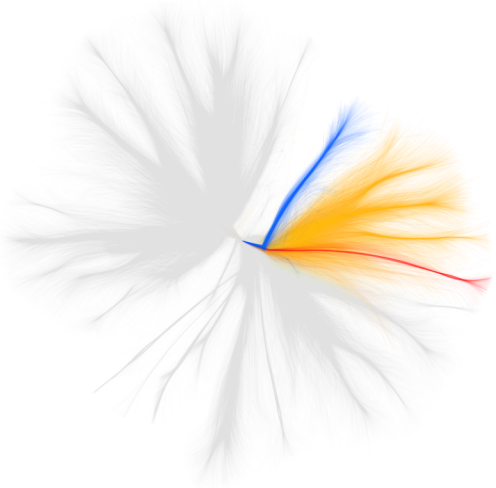
How much confidence do you have in your answer? *

Low Confidence 1 2 3 4 5 High Confidence

How much effort was it to come to the answer? *

Little Effort 1 2 3 4 5 Much Effort

Next



(j) Question 10

Task:
Look at the visualization provided and answer the following question:

Which of the following behaviour can you identify among the three groups, assuming there are 2 phases (earlier, later) that equally divide the optimization? *

- They share trajectories before they split at the earlier phase and in the end, the majority of the pink group is more related to the blue group than the orange group
- They share trajectories before they split at the later phase and in the end, the majority of the pink group is more related to the blue group than the orange group
- They share trajectories before they split at the earlier phase and in the end, the majority of the pink group is more related to the orange group than the blue group
- They share trajectories before they split at the later phase and in the end, the majority of the pink group is more related to the orange group than the blue group
- I cannot answer

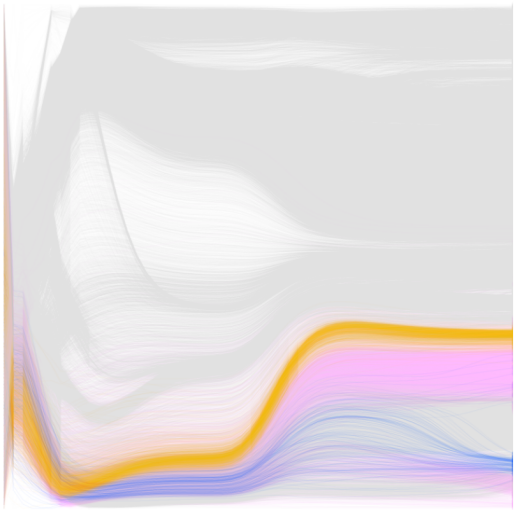
How much confidence do you have in your answer? *

Low Confidence 1 2 3 4 5 High Confidence

How much effort was it to come to the answer? *

Little Effort 1 2 3 4 5 Much Effort

Next



(k) Question 11

Task:
Look at the visualization provided and answer the following question:

Which of the following behaviour can you identify among the three groups, assuming there are 2 phases (earlier, later) that equally divide all iterations *

- They share trajectories before they split at the earlier phase and in the end, the the majority of the orange group is closer to the blue group than the magenta group
- They share trajectories before they split at the later phase and in the end, the majority of the orange group is closer to the blue group than the magenta group
- They share trajectories before they split at the earlier phase and in the end, the majority of the orange group is closer to the magenta group than the blue group
- They share trajectories before they split at the later phase and in the end, the majority of the orange group is closer to the magenta group than the blue group
- I cannot answer

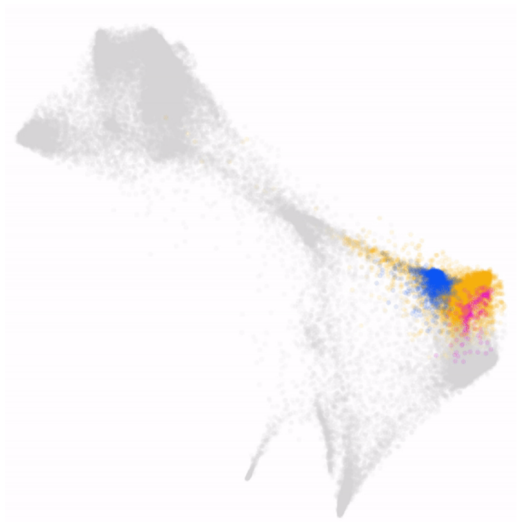
How much confidence do you have in your answer? *

Low Confidence 1 2 3 4 5 High Confidence

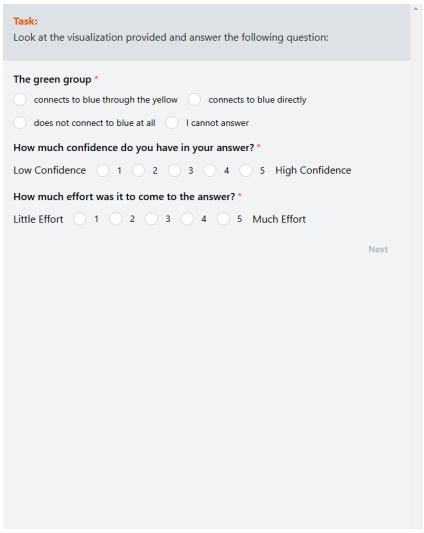
How much effort was it to come to the answer? *

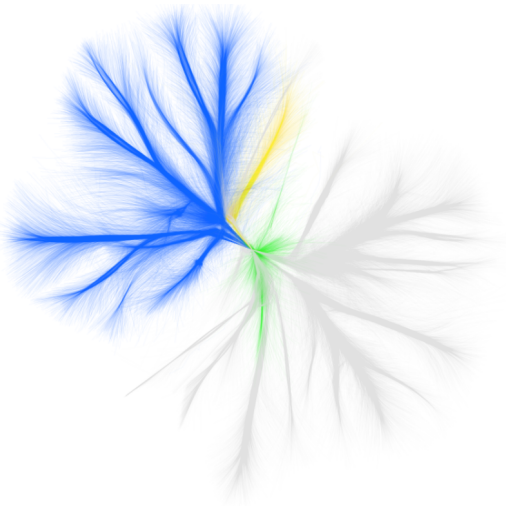
Little Effort 1 2 3 4 5 Much Effort

Next

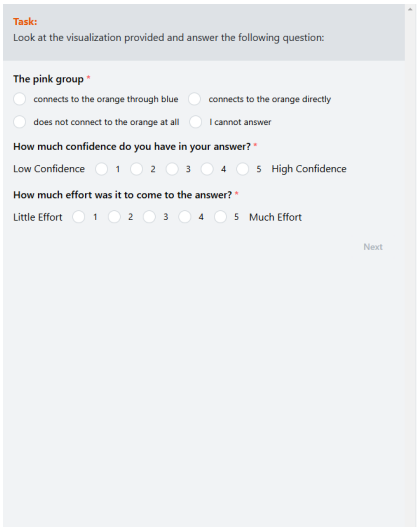


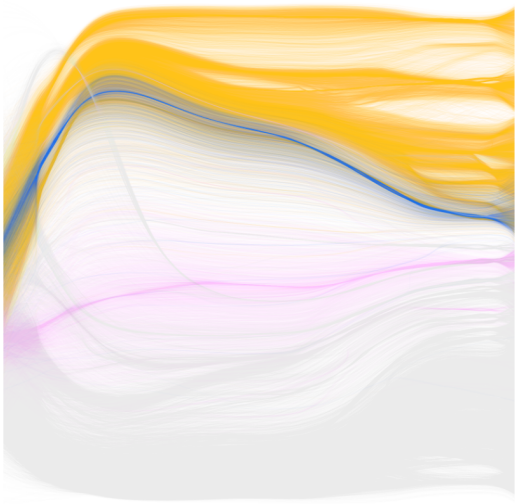
(l) Question 12



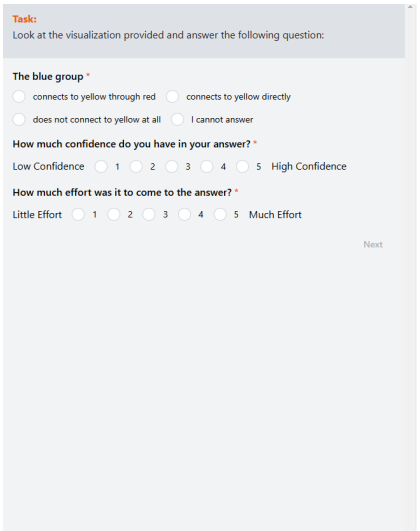


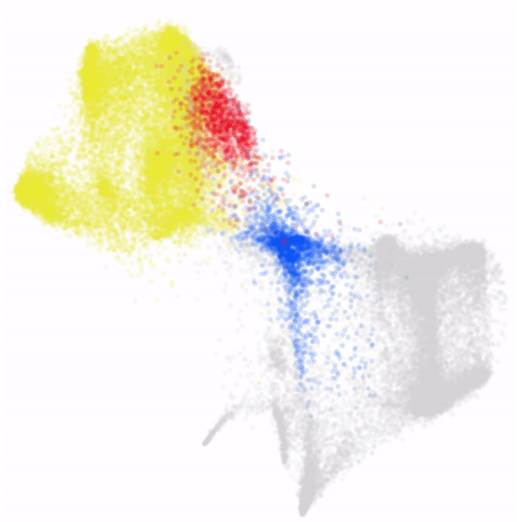
(m) Question 13



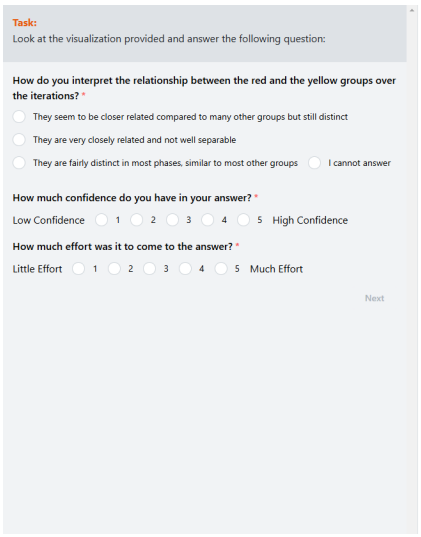


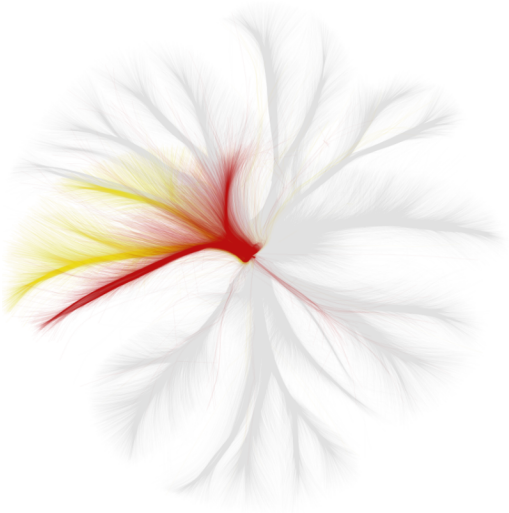
(n) Question 14



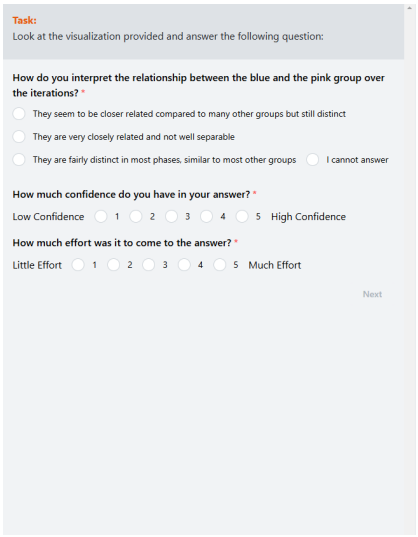


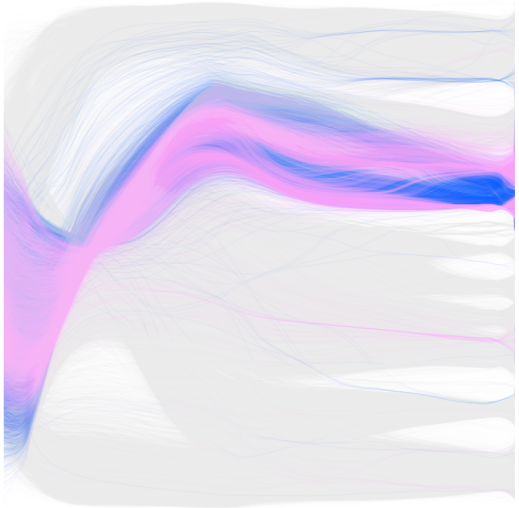
(o) Question 15



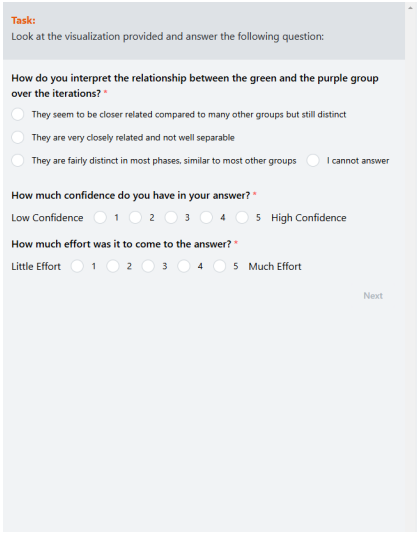


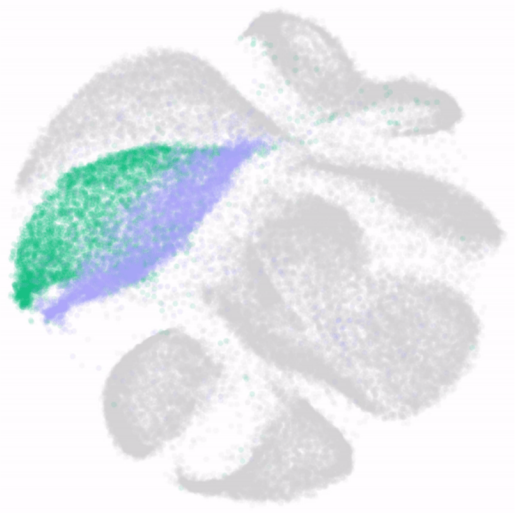
(p) Question 16



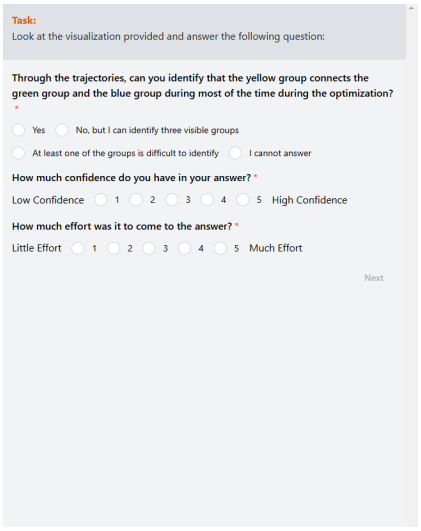


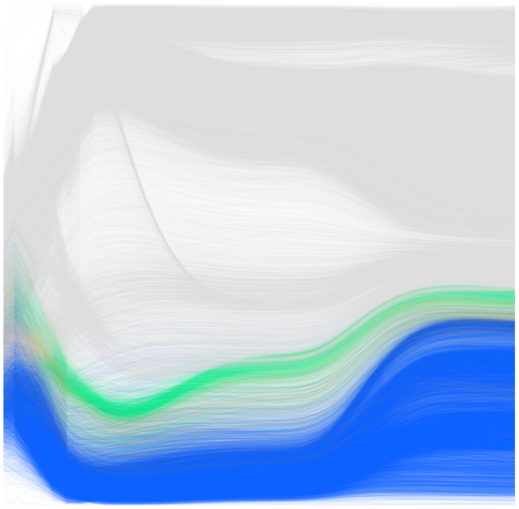
(q) Question 17



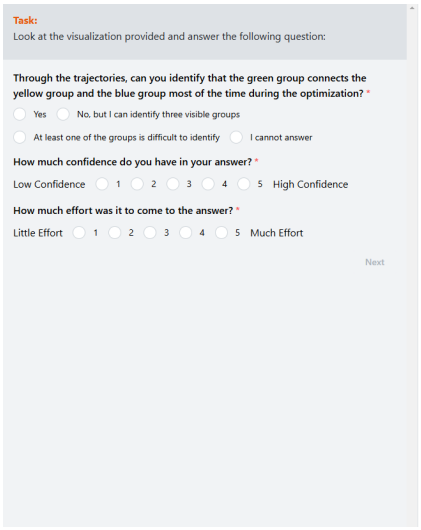


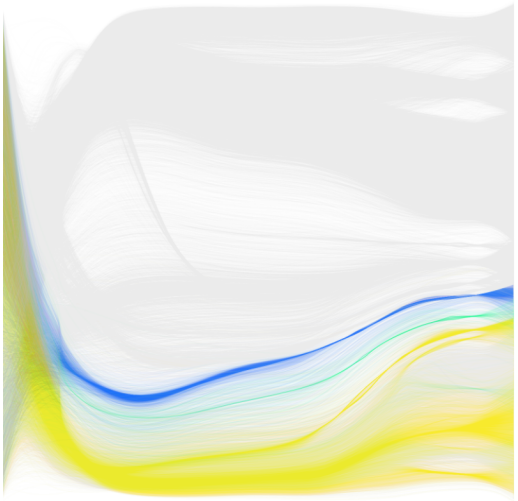
(r) Question 18



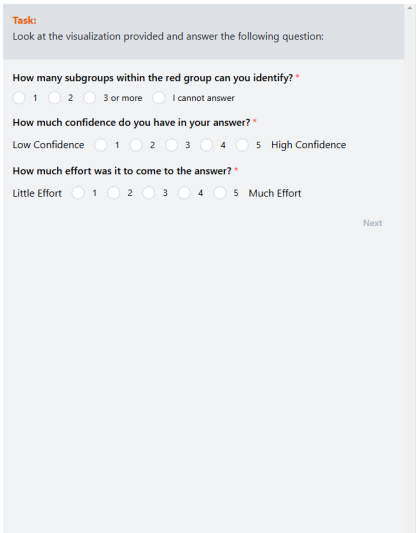


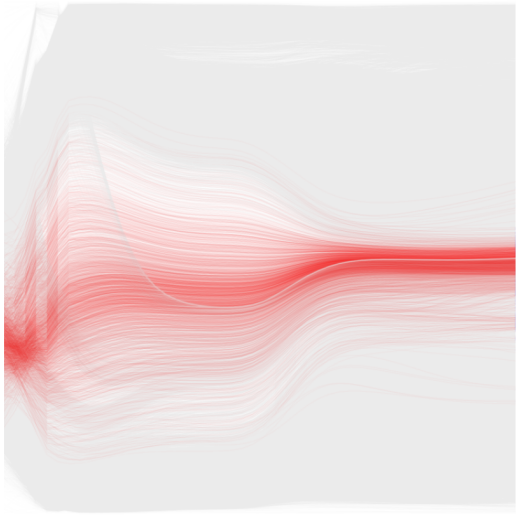
(s) Question 19



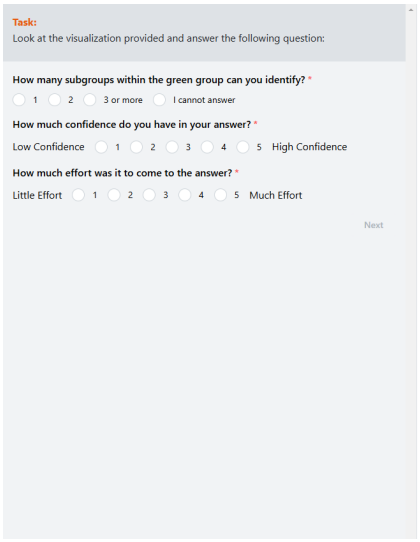


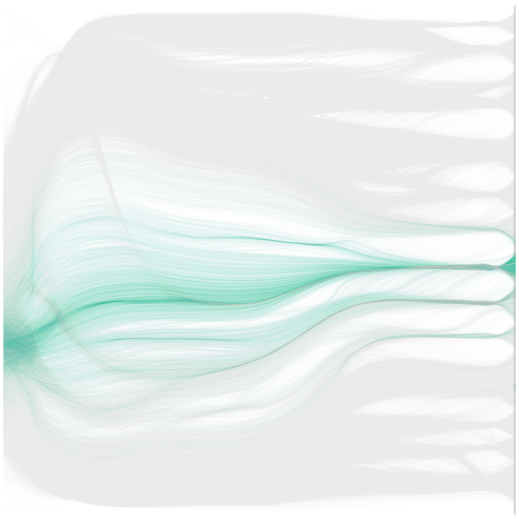
(t) Question 20



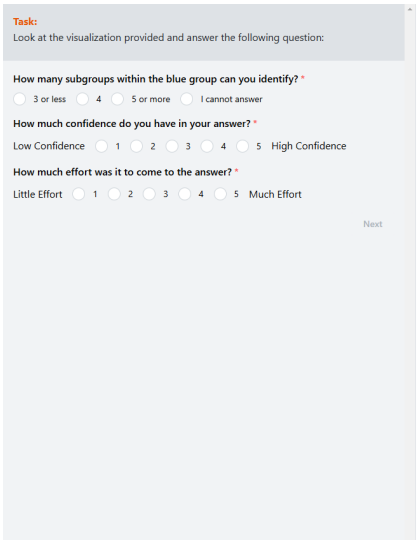


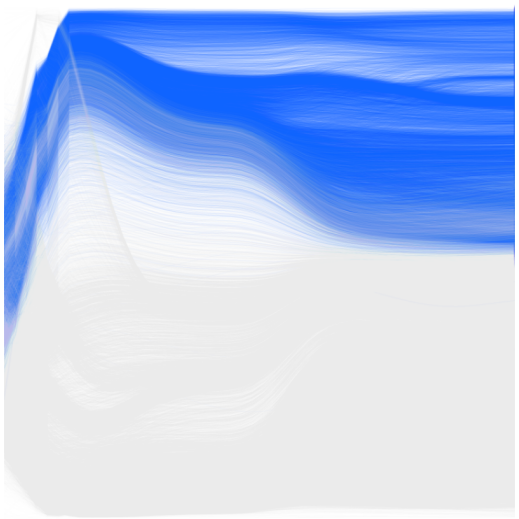
(u) Question 21



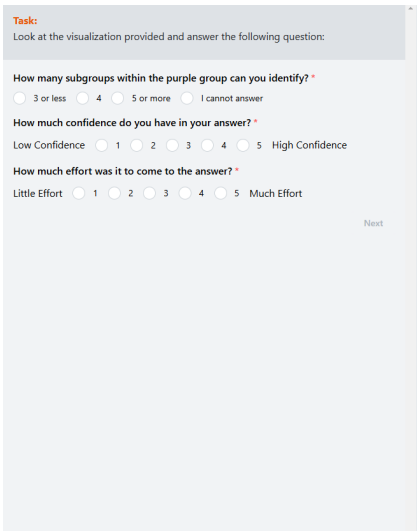


(v) Question 22

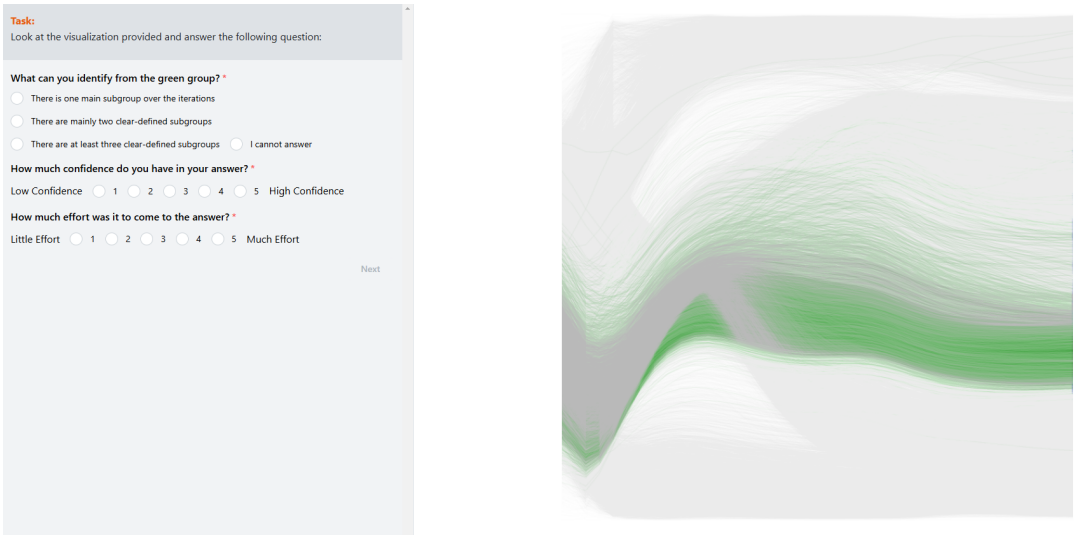




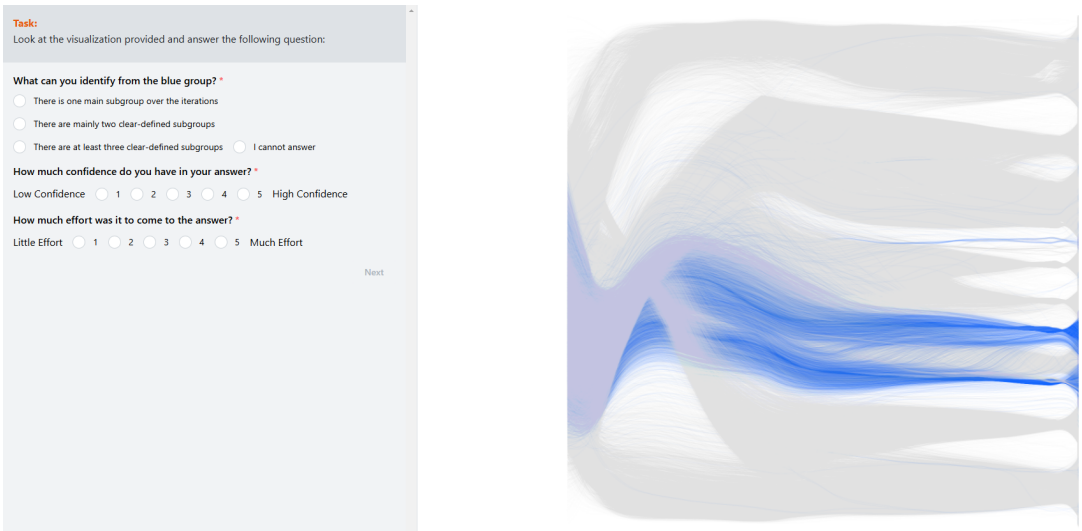
(w) Question 23



(x) Question 24



(y) Question 25



(z) Question 26

Figure A.1: Questions asked during the general user test. In the real tests, question 3, 6, 9, 12, 15 and 18 are animated. Questions were provided to participants in random order.

Question Groups	Individual Question IDs
Group 1	Question 1, 2, 3
Group 2	Question 4, 5, 6
Group 3	Question 7, 8, 9
Group 4	Question 10, 11, 12
Group 5	Question 13, 14, 15
Group 6	Question 16, 17, 18
Group 7	Question 19, 20
Group 8	Question 21, 22
Group 9	Question 23, 24
Group 10	Question 25, 26

Table A.4: Mapping between question groups and IDs of individual questions