

Optimizing Stepwise Animation in Dynamic Set Diagrams

Kazuyo Mizuno¹, Hsiang-Yun Wu², Shigeo Takahashi³, and Takeo Igarashi⁴

¹Yahoo Japan Corporation, Japan

²TU Wien, Austria

³The University of Aizu, Japan

⁴The University of Tokyo, Japan

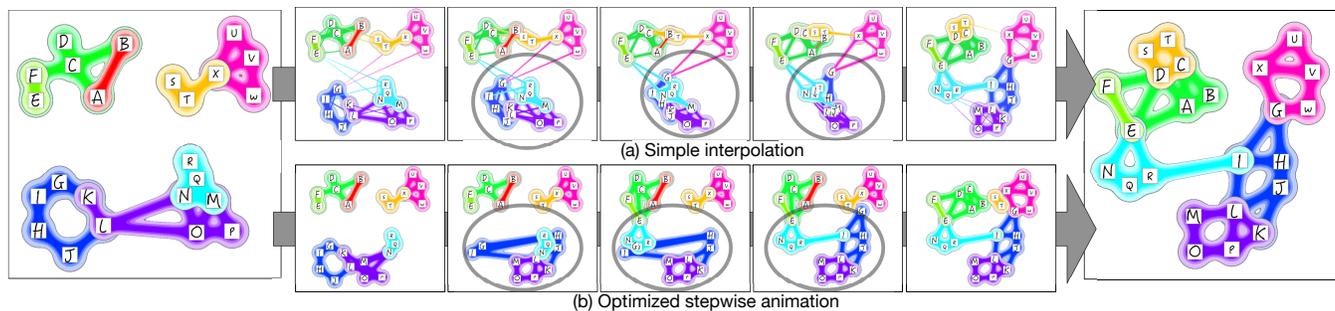


Figure 1: Side-by-side comparison of the interpolation method between two set diagrams. (a) A simple interpolation in which elements are linearly translated simultaneously. (b) Our optimized stepwise animation in which the transition is decomposed to a sorted sequence of atomic changes. Each element is represented as an image of the alphabet and each set is represented as a colored region.

Abstract

A set diagram represents the membership relation among data elements. It is often visualized as secondary information on top of primary information, such as the spatial positions of elements on maps and charts. Visualizing the temporal evolution of such set diagrams as well as their primary features is quite important; however, conventional approaches have only focused on the temporal behavior of the primary features and do not provide an effective means to highlight notable transitions within the set relationships. This paper presents an approach for generating a stepwise animation between set diagrams by decomposing the entire transition into atomic changes associated with individual data elements. The key idea behind our approach is to optimize the ordering of the atomic changes such that the synthesized animation minimizes unwanted set occlusions by considering their depth ordering and reduces the gaze shift between two consecutive stepwise changes. Experimental results and a user study demonstrate that the proposed approach effectively facilitates the visual identification of the detailed transitions inherent in dynamic set diagrams.

CCS Concepts

• **Human-centered computing** → **Information visualization**; *Graph drawings*; • **Computing methodologies** → *Perception*;

1. Introduction

Visualization of dynamic data helps us perceive important characteristics of time-varying information. In practice, it is a technically challenging problem that researchers in the visualization community have attempted to address. Typically, the visualization of dynamic data comprises two technical problems: a correspondence problem and an interpolation problem. The correspondence problem involves extracting meaningful correspondences from multiple

temporal snapshots of dynamic data. This allows us to find a reasonable layout of the next temporal snapshot by referring to the previous snapshot while preserving the associated mental map. In contrast, the interpolation problem involves finding a perceptually plausible transformation between data in adjacent temporal snapshots.

The visualization of dynamic set diagrams is beneficial because it allows us to understand the temporal changes in set memberships

associated with data elements. However, only a few studies have investigated this problem, despite the existence of real dynamic set data. This is because it is necessary to manage the memberships of data elements as secondary information as well as primary information, for example, spatial positions that are often predefined in visualization images, such as geographical maps and statistical charts. Even if the positions of data elements are not fully constrained, we cannot arbitrarily compose a new set diagram layout by referring to a previous temporal snapshot. This is because set diagrams are often composed in such a way that the data elements in the same set are close to each other. This is done to maximally suppress unwanted conflicts among set contours for better readability. In this case, the correspondence problem can be solved naturally by understanding the set memberships of the data elements in the diagrams. However, the interpolation problem becomes significantly complex because we often have to transform between two distinct layouts of the data elements by referring to the underlying changes in their set memberships. In Figure 1(a), a simple interpolation scheme is employed to visualize the transformation between two set diagrams. Unfortunately, we cannot clearly identify the details of changes in the set memberships of the data elements in this case since all temporal transitions are made in a single step. Moreover, dynamic set diagrams inevitably suffer from occlusions of the sets and data elements, which degrades the clarity of the synthesized animation. Thus, the visualization of dynamic set diagrams differs from that of dynamic graphs and consequently requires its own specific solution.

In this paper, we present an animation approach for such dynamic set diagrams to fully elucidate the underlying temporal changes in the set memberships of the data elements. For this purpose, we employed a graph-based representation of a set diagram, which helps decompose the entire temporal transition into atomic changes associated with the data elements for composing stepwise animations automatically. Our technical contribution lies in optimizing the sequence of the atomic changes to reduce the gaze shifts imposed on the viewers and in arranging the depth ordering of sets during the animation to minimize occlusions. In addition, we decrease the duration of the entire transition by clustering conflict-free atomic changes while retaining visual clarity in the animation. Figure 1(b) shows a stepwise animation between two set diagrams generated automatically using our approach. Compared to the simple interpolation on the top Figure 1(a), our approach can successfully visualize how the source set diagram changes into the target one while suppressing occlusions among the sets.

Note that our optimization criteria for reducing the cognitive load originate from the study performed by Bauhoff et al. [BHS12], where the researchers claimed that working memory use increases when the gaze shift becomes longer. Since our approach expects users to focus their visual attention on the series of atomic changes that constitute the difference between the two set diagrams, we want to minimize the total distance of such gaze motion during the stepwise animation to fully reduce the associated cognitive load.

The remainder of this paper is organized as follows. Section 2 provides a survey of related work, including visualization techniques for static set diagrams and dynamic graphs. Section 3 overviews the proposed approach and the notations used in this pa-

per are described, and Section 4 includes our graph representations associated with set diagrams. Section 5 presents the primary contributions of this work by describing our techniques to compose a stepwise animation of a dynamic set diagram. Experimental results, the evaluation with user studies, and discussions are given in Section 6. Section 7 presents conclusions and future work.

2. Related Work

In this section, we investigate the most relevant techniques for visualizing set diagrams and dynamic graphs.

2.1. Set Diagram Visualization

The visualization of set memberships has been extensively studied in the information visualization community, which has resulted in a wide range of proposed visualization algorithms [AMA*14]. Venn and Euler diagrams are classical representations of set-typed data, where associated data elements are arranged such that they are in close proximity if they belong to the same set. An individual set is typically bounded by a closed curve called a *contour*, and an intersection among multiple sets is represented by their overlap. Various techniques have been developed to automate the layout design and color assignment of Euler diagrams [Rod14, SA08, RD10]. However, conventional techniques for drawing Euler diagrams and their variants are limited to static cases, and thus they fail to directly convey details about the dynamic transitions of sets.

Several previous approaches have been used for such set diagrams that are overlaid on preexisting visualization images, such as statistical charts and geographical maps. *Bubble Sets* [CPC09] introduced an energy field to outline the shape of each set on the underlying image by employing an implicit surface representation. Meanwhile, colored textures are applied to render areas of interest to visualize multiple metrics associated with constituent sets together with the relationships among them [BT09]. *Line-Sets* [ARRC11] and its successor *KelpDiagram* [DvKSW12] employed continuous lines and sparse networks, respectively, to connect data elements in the same set. Recently, a hybrid of such line- and region-based set representations has been further improved as *KelpFusion* [MRS*13].

In contrast, few studies have investigated dynamic set visualization. Rodgers et al. [RMF04] proposed a technique for rearranging Euler diagrams by incorporating the layouts of reference graphs to preserve users' mental maps. To improve user intervention, approaches have been developed to explore annotated modules in biological networks by extending self-organizing maps [DEKB*14] and to edit Venn diagrams, which offers a way to analyze set unions while preserving the overall shape of the diagram [HMdS*15]. Bremm et al. [BLA*11] extended the concept of *Parallel Sets* to analyze time-dependent group memberships to interactively identify meaningful trends within time-varying data. These approaches have successfully employed interactions on set diagrams using predefined diagrams or networks, but it is not designed to analyze detailed time-dependent changes in the set structure.

In this paper, we tackle the problem of dynamic set visualization. Our focus is on visually elucidating individual changes in the

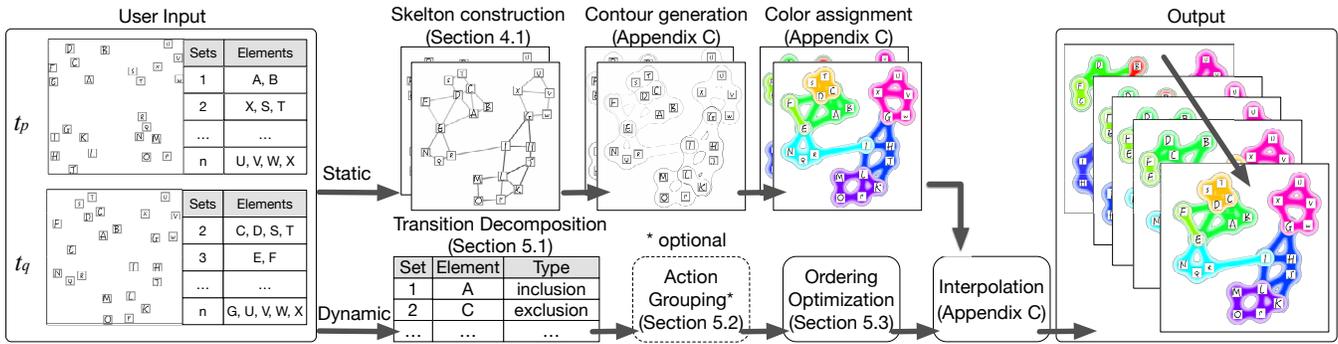


Figure 2: Overview of the proposed approach. The asterisk “*” indicates an optional step.

set membership while maximizing temporal coherence in the visualization of the dynamic set diagrams.

2.2. Dynamic Graph Visualization

In regard to time-varying data, the visualization of dynamic graphs is not a new research area [Nor96]; it has been studied aggressively in recent years [BBDW14]. Temporal coherence in the visualization of dynamic graphs that allows viewers to maintain their mental map is a primary technical challenge, which has been discussed in depth by Archambault et al. [AP13]. As described earlier, this technical challenge leads to *correspondence* and *interpolation* problems, particularly when handling offline dynamic graphs.

To solve the above two key problems, Brandes et al. [BIM12] developed aggregation and anchoring techniques to handle dynamics, under the assumption that the networks have relatively constant global structures. The animation of a dynamic graph has also been compared to its multiple temporal snapshots, and Archambault et al. [APP11, AP16] concluded that animation is more useful when accuracy is more important than speed. The pioneering work of Friedrich et al. [FE01, FH02] decomposed a temporal transition of a dynamic graph into components. Rufiange and McGuffin [RM13] subsequently implemented *DiffAni* to present differences in multiple graphs or animations between temporal snapshots according to user preferences. The concept was further developed by Bach et al. [BPF14], whose system, *GraphDiaries*, depicts graph dynamics through a staged animated transition. Moreover, to advance the visualization of dynamic graphs, they are often displayed with clustered node structures [FA04, LSCL10, MKY12].

We consider that the most crucial gap between the problems of visualizing dynamic set diagrams and dynamic graphs is that we have to maximally avoid occlusion of individual sets. The dynamic set diagrams inherently suffer from this occlusion problem because they generally employ region-based representations unlike graphs consisting of nodes and links only. In this work, we address this gap by composing stepwise animations to visually analyze offline dynamic sets while maximizing the visibility of evolving sets and the temporal coherence in their relationships.

3. Visualization Design

Before describing our approach in detail, we summarize the visualization design criteria employed in the proposed approach.

3.1. Overview

In our approach, we take as input a given collection of data elements $M = \{m_1, m_2, \dots, m_n\}$, each of which is associated with multiple sets $S = \{s_1, s_2, \dots, s_m\}$. We assume that the individual elements and sets have unique IDs and that the 2D positions of the elements at specific times are given beforehand. Each set is expected to contain at least one element and have some overlap with other sets. Every set has a unique color assigned by the system as described in the supplementary material.

Suppose that each temporal snapshot T_i at time t_i consists of data elements $M_i = \{m_{i1}, m_{i2}, \dots, m_{in}\}$ and constituent sets $S_i = \{s_{i1}, s_{i2}, \dots, s_{im}\}$. Note that m_{ij} ($j = 1, \dots, n$) also represents the 2D positions in the set diagram throughout this paper. Our task is to find a proper interpolation between a pair of snapshots $T_p(M_p, S_p)$ and $T_q(M_q, S_q)$, where T_p and T_q correspond to a source and target set diagram, respectively. Note that the elements and sets can appear/disappear during a temporal change in the dynamic set diagram according to whether they constitute the entire diagram. Our visualization goal is to clarify the individual differences in the set membership of the data elements between a specific pair of set diagrams T_p and T_q while retaining the static set relationships in the two temporal snapshots. As mentioned previously, we assume that the positions of the data elements are predefined in the given diagrams. This means that we only focus on the interpolation problem for the dynamic set diagrams in this paper and thus exclude the correspondence problem for finding the next layout of data elements from our consideration.

Figure 2 presents an overview of the proposed approach. The top path in the figure illustrates the process of rendering static set diagrams, and the bottom path corresponds to the process of composing a stepwise animation for dynamic set diagrams. In the visualization of static set diagrams, we display set relationships over the pre-existing elements by associating graph structures with the sets so that we can outline the profile of each set as a contour line. In contrast, in the visualization of dynamic set diagrams, we decompose

the temporal transition of the set membership into atomic changes and sort them to generate a stepwise animation while maintaining the temporal coherence of the graph structures associated with the sets. Note that we call the decomposed atomic change an *action*.

3.2. Visualization Design Criteria

The goal of this study is to clearly visualize how each data element moves or participates in and/or leaves specific sets during the transition between a pair of consecutive set diagrams to help users reliably follow these individual actions. In the proposed approach, we achieve this by generating stepwise animations that fulfill the following design criteria.

- C.1 Better visibility:** The visibility of each action should be enhanced during the stepwise animation, in particular by suppressing undesirable occlusions and unrelated visual effects.
- C.2 Limited degree of complexity:** Each stepwise transition is forced to contain only a few actions to effectively assist viewers to visually interpret the target action by following the visualization design proposed by Bach et al. [BPF14].
- C.3 Minimal gaze shift:** Total gaze shift throughout the animation should be minimized to reduce the cognitive load imposed on viewers [BHS12].

4. Rendering Static Set Diagrams

First, we explain how we render a set diagram at each time step. As a variant of the line-based representation as described in Section 2.1, the proposed approach constructs a skeleton graph as a basis for representing each set since this representation can delineate its topological transition easily compared to a region-based representation. In the rendering process, we perform the following three steps for each set, while steps 2 and 3 are described in the supplementary material.

1. *Skeleton construction:* We construct a skeleton graph for each set by referring to the positions of its member elements.
2. *Contour generation:* We dilate the skeleton graph to generate a closed region and extract its profile as a contour.
3. *Color assignment:* We assign a unique color to the set and apply its smooth gradation to the inside of the corresponding region.

Skeleton Construction. In the proposed approach, we use a *shortest-path graph* as the skeleton of each set. The shortest-path graph, which was proposed by de Berg et al. [dBMS11], can capture the spatial distribution of elements as a graph with variable degrees of detail. In practice, the shortest-path graph can control the sparseness of the graph, which allows us to explore the balance between line-based and region-based representations. We have extended their shortest-path graph construction process to maintain consistency between temporal snapshots.

According to *KelpFusion* [MRS^{*}13], the construction of the shortest-path graph begins with the generation of a Delaunay triangulation with the given positions of the collection of data elements in the target set. We then construct the shortest-path graph by selecting edges from those of the Delaunay triangulation. Note that, in the original *KelpFusion* process, edges for the shortest-path graph are selected from the Delaunay triangulation in the order of

edge length. However, we want to maximally maintain the consistency of edge connectivity with respect to the temporal change in set membership to establish criteria C.1 and C.3 (in Section 3.2). Therefore, we modify the priority of selecting the edges of the shortest-path graph by referring to the positions of the elements and their set memberships in the set diagrams at two time samples.

Suppose that we consider the transition between the two diagrams T_p and T_q at time samples t_p and t_q , respectively. In our implementation, we first compute the Delaunay triangulation of each set in that order and assign default priority values to the respective edges in the triangulation by referring to their length. However, we also adjust the edge priority values for our specific purpose by employing several criteria. Suppose that we denote the Delaunay triangulations of s_{pj} and s_{qj} by D_{pj} and D_{qj} , respectively. We can summarize the criteria as follows (Figure 3):

- *Local temporal coherence* (Figure 3(a)): We suppress the unnecessary disappearance of the edge during the transformation. Suppose that an identical edge is shared by D_{pj} and D_{qj} . We suppress the unnecessary disappearance of this edge during the transformation if the displacements of its two end elements m_1 and m_2 from T_p to T_q are both less than a given threshold. In this case, we raise the priority of this edge in such a way that the edge is more likely to be chosen in the corresponding shortest-path graph.
- *Global temporal coherence* (Figure 3(b)): We retain the presence of this edge in the corresponding shortest-path graph at t_q if it connects two specific elements m_1 and m_2 in both D_{pj} and D_{qj} to maintain the temporal coherence of this edge between the two temporal snapshots. This is accomplished by giving higher priority to such an edge before extracting the shortest-path graph from the Delaunay triangulation D_{qj} .
- *Set connectivity* (Figure 3(c)): We suppress an unwanted inconsistency in the connectivity of the shortest-path graphs of two different sets s_{ik} and s_{ij} at t_i ($i = p, q$). When an identical edge is shared by the corresponding Delaunay triangulations D_{ik} and D_{ij} , we preserve the existence of this edge if its end elements (e.g., m_2 and m_3) are shared by the two sets or more at each time sample.

Note that, in our formulation, we apply these three criteria to the Delaunay edges to construct the shortest-path graphs.

5. Composing Optimized Stepwise Animations

The primary contribution of this work is described in this section; that is, generating stepwise animations to clarify individual changes inherent in a dynamic set diagram. We synthesize stepwise animations similar to those developed in *GraphDiaries* [BPF14]; however, our technical challenge is further complicated because we must clarify individual changes in the set membership of the data elements in addition to their displacements. This problem is solved by respecting the design criteria described in Section 3.2, where C.2 provides guidelines for decomposing the entire transition between two different set diagrams (Section 5.1), and C.3 and C.1 allow us to formulate an optimization problem to rearrange the ordering of the actions in the stepwise animation (Section 5.3). In practice, the entire sequence of the stepwise animation requires considerable time to preview if it contains a large number of actions.

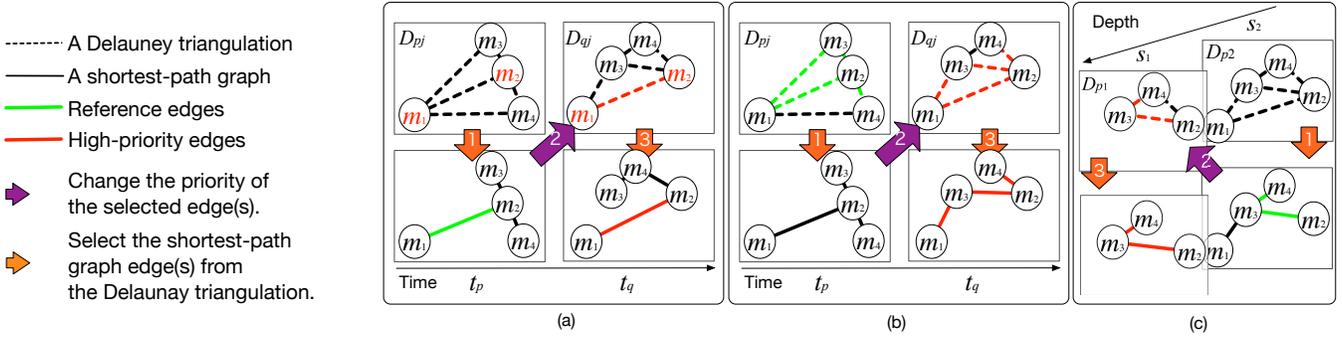


Figure 3: Adjusting the priority values of edges in the Delaunay triangulation of each set for our specific purpose. This allows us to compose a shortest-path graph for each set such that the graph respects the following three criteria: (a) local temporal coherence, (b) global temporal coherence, and (c) set connectivity.

Thus, when required, we add an action grouping process prior to re-arranging the ordering of actions to control the balance between visualization clarity and the number of animation steps (Section 5.2). Consequently, in the process of generating optimized stepwise animations, we perform the following three steps:

1. *Decomposition of temporal transitions:* We decompose the entire transition between two set diagrams into actions.
2. *Grouping actions into clusters* (optional): We group compatible actions into clusters while preserving visualization clarity.
3. *Optimization of the action cluster order:* We optimize the ordering of action clusters in consideration of the design criteria.

5.1. Temporal Transition Decomposition

According to C.2, we first decompose the entire transition between temporal snapshots $T_p(M_p, S_p)$ and $T_q(M_q, S_q)$ into actions a_1, a_2, \dots , and a_k , where each action can be represented as follows:

$$a_i = (m_{M(i)}^p, m_{M(i)}^q, s_{S(i)}, c_i) \quad i = 1, 2, \dots, k, \quad (1)$$

where k is the number of actions. Here, the action consists of the source and target positions of elements $m_{M(i)}^p$ and $m_{M(i)}^q$ at t_p and t_q , respectively; target set $s_{S(i)}$, and action type c_i . $M(i)$ and $S(i)$ are the IDs of the elements and sets associated with the i -th action a_i , respectively. For convenience, we categorize actions (during $[t_p, t_q]$) into the following five types (Figure 4):

- A.1 Disappearance:** An existing element disappears from a set.
- A.2 Appearance:** A new element appears in a set.
- A.3 Exclusion:** An element leaves a set.
- A.4 Inclusion:** An existing element joins a new set.
- A.5 Translation:** An element simply moves while retaining its set membership.

5.2. Grouping of Actions into Clusters

We then optionally group compatible actions into clusters to reduce the total number of animation steps, which makes it possible to seek a compromise between visualization clarity and the length of the animation. In this approach, we first collect all the possible groups of actions that can coexist into a single animation step

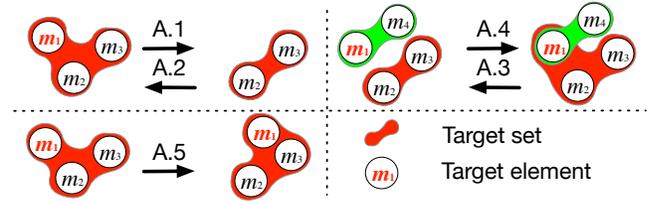


Figure 4: Five types of actions in our approach.

and then determine the optimal subsets of actions that cover the entire collection of actions without any overlap. Our conditions for grouping actions a_i and a_j are summarized as follows:

- *Spatial closeness:* Elements $m_{M(i)}^p$ and $m_{M(j)}^p$ are sufficiently close to each other in T_p , or elements $m_{M(i)}^q$ and $m_{M(j)}^q$ are sufficiently close to each other in T_q (Figure 5(a)).
- *Motion similarity:* The translation direction of $m_{M(i)}$ and $m_{M(j)}$ is sufficiently similar, on the condition that c_i and c_j are translation type (A.5) within the same set (Figure 5(b)).
- *Action consistency:* Actions a_i and a_j must satisfy at least one of the conditions shown in Figure 6. Generally, we group actions of the same type, which are associated with the same element (Figure 6(a), (c), and (e)) or the same set (Figure 6(b), (d), and (f)). We also accept a pair with A.4 (or A.2) and A.3 (or A.1) associated with a single set (Figure 6(g)). However, we do not allow the same pair if it influences multiple sets during a single animation step (Figure 6(h)), as this is more likely to cause unwanted occlusion among the multiple sets. Thus, we eliminated this case from the list of acceptable action pairs.

To find an optimal collection of subsets of actions, we solve a variant of the *set cover problem* that finds an optimal collection of subsets that covers all elements while minimizing a predefined cost function as described in the supplementary material.

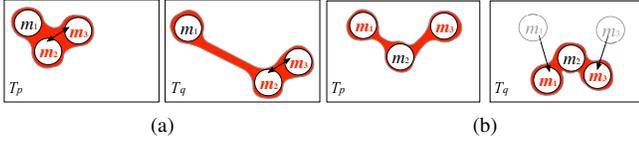


Figure 5: Spatial and motion conditions for grouping actions as a cluster: (a) spatial closeness and (b) motion similarity (clustered actions are indicated by the red font).

5.3. Optimization of the Action-Cluster Order

To fully satisfy criteria **C.3** and **C.1**, we optimally sort the clusters of actions (Section 5.2) in the entire animation. Suppose that we have k' action clusters, each of which consists of a single action or a cluster of actions as shown in Figure 7. Here, we have to search for an optimized permutation of such action cluster IDs $\{1, \dots, k'\}$ to generate a visually appealing stepwise animation. This is accomplished by seeking a route that travels all positions of these action clusters exactly once while minimizing the total cost of the route. Thus, we formulate our optimization problem for ordering action clusters, as a variant of the traveling salesman problem and search for the permutation that minimizes the following cost function:

$$\sum_{\sigma(i)=1}^{k'} (\alpha F_1(g_{\sigma(i)}) + \beta F_2(g_{\sigma(i)}) + \gamma F_3(g_{\sigma(i)}) + \delta F_4(g_{\sigma(i)})), \quad (2)$$

where σ is a permutation of the action cluster IDs $\{1, \dots, k'\}$ to be optimized and $g_{\sigma(i)}$ is the i -th action cluster in σ . Note that F_1 , F_2 , F_3 , and F_4 are cost functions that penalize the total gaze shift, visual clutter arising from the order of action clusters, frequency of changes in the depth order of sets, and existence of multiple components in skeleton graphs, respectively. In the following, the four cost functions are explained in detail.

Total Gaze Shift. To incorporate criterion **C.3**, we define cost function F_1 to evaluate the total amount of gaze shift. In practice, we sum up the distances between two consecutive action clusters. If the type of the current action cluster is translation (**A.5**), we also include the associated displacement of the action cluster. Thus, we can write

$$F_1(g_{\sigma(i)}) = |P'_{\sigma(i)} - P_{\sigma(i)}| + |P_{\sigma(i+1)} - P'_{\sigma(i)}|, \quad (3)$$

where $P_{\sigma(i)}$ is the gravity center of data elements in action cluster $g_{\sigma(i)}$, and $P'_{\sigma(i)}$ corresponds to the center of the data elements after all translations of these elements are carried out. Note that $P'_{\sigma(i)} = P_{\sigma(i)}$ if $g_{\sigma(i)}$ is a non-translation action cluster.

Visual Clutter Arising from the Order of Action Clusters. To maximally respect criterion **C.1**, we introduce three rules for ordering the action clusters by referring to their types as follows.

- *Priority order of all actions:* The observations in *GraphDiaris* [BPF14] suggest that the number of displayed elements and edges should be reduced as much as possible to simplify the information contained in the static images. By following this idea, we prefer to order actions of the disappearance (**A.1**) and exclusion (**A.3**) types first, then the translation (**A.5**) type, and finally the appearance (**A.2**) and inclusion (**A.4**) types.

- *Priority order among disappearance and exclusion actions:* Actions of the disappearance (**A.1**) and exclusion (**A.3**) types shrink the sizes of the associated sets, and thus smaller sets should be handled earlier in the sequence of such actions to maximally expose changes in the set diagram (Figure 8).
- *Priority order among appearance and inclusion actions:* In contrast, actions of the appearance (**A.2**) and inclusion (**A.4**) types should emerge in descending order of size (Figure 8).

Here, to penalize pairs of action clusters that violate the above rules, we formulate the cost function F_2 as follows:

$$F_2(g_{\sigma(i)}) = \sum_{j=i+1}^{k'} L(g_{\sigma(i)}, g_{\sigma(j)}) \quad (4)$$

$$L(g_{\sigma(i)}, g_{\sigma(j)}) = \sum_{(a_{i'}, a_{j'}) \in G} L'(a_{i'}, a_{j'}) \quad (5)$$

$$G = g_{\sigma(i)} \times g_{\sigma(j)} = \{(a_{i'}, a_{j'}) | a_{i'} \in g_{\sigma(i)} \wedge a_{j'} \in g_{\sigma(j)}\} \quad (6)$$

$$L'(a_{i'}, a_{j'}) = \begin{cases} 1 & \text{if } O(a_{i'}) > O(a_{j'}), \\ 1 & \text{if } O(a_{i'}) = O(a_{j'}) = 0 \text{ and } D(s_{S(i')}) > D(s_{S(j')}), \\ 1 & \text{if } O(a_{i'}) = O(a_{j'}) = 2 \text{ and } D(s_{S(i')}) < D(s_{S(j')}), \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

where $O(a_{i'})$ returns 0, 1, or 2 if $a_{i'}$ is of the disappearance/exclusion, translation, or appearance/inclusion types, respectively. In addition, $D(s_{S(i')})$ indicates the depth order of set $s_{S(i')}$ prior to performing $g_{\sigma(i)}$ and decreases if the set approaches the top of the depth layers. For example, let us find the optimal sequence of actions that transforms the leftmost set diagram to the rightmost one in Figure 8. In this case, $L'(a_{R-}, a_{G-}) = L'(a_{B-}, a_{G-}) = L'(a_{B-}, a_{R-}) = 1$ while $L' = 0$ otherwise (Eq. (7)). This is because all these actions are of the exclusion type and the green, red, and blue sets are arranged in this depth order from top to bottom in the diagram. Thus, the optimal sequence is a_{G-} , a_{R-} , and a_{B-} among the possible action sequences.

Frequency of Changes in the Depth Order of Sets. It is possible to swap sets in the depth order when the number of elements in a certain set changes according to some action. This inevitably causes additional unwanted visual clutter, which violates the criterion **C.1**. Thus, we aim to penalize such changes in the depth order of sets by introducing the cost function F_3 as the number of sets included in T_p and T_q , which changes its depth after performing $g_{\sigma(i)}$.

Existence of Multiple Components in Skeleton Graphs. It is still possible that a skeleton graph unnecessarily splits and merges during an animation step, which causes unwanted confusion, especially for viewers who try to identify meaningful changes in the set diagram. We aim to penalize such unwanted separation of each set by introducing the cost function F_4 as the number of connected components in the skeleton graph of sets included in T_p and T_q .

6. Validation

We performed three different types of experiments to assess the capability of the proposed approach, which can be summarized as:

- *Computational performance:* This experiment was conducted for evaluating the performance of our approach (Section 6.2).

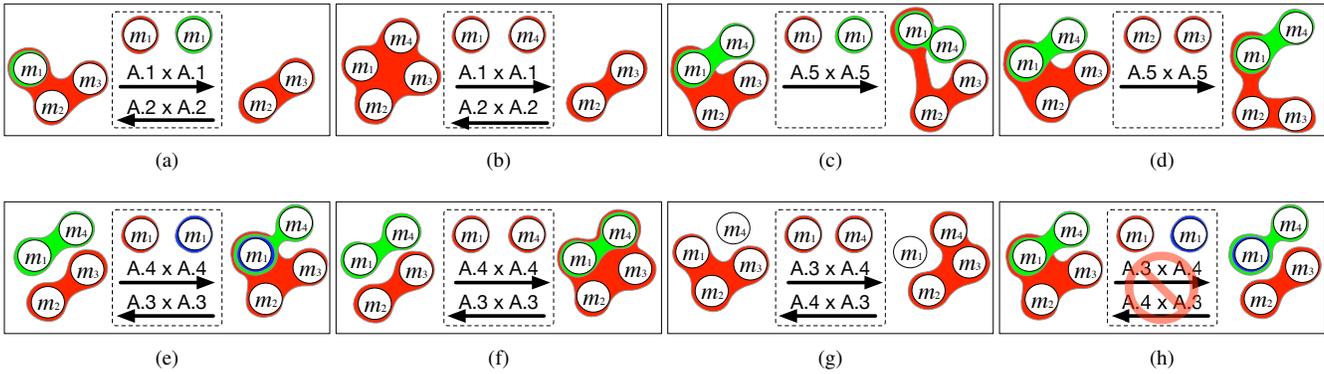


Figure 6: Conditions for grouping actions as a cluster that corresponds to a single animation step. Appearance/Disappearance actions are applied to (a) an element of (b) a set. Translation actions are applied to (c) an element or (d) a set. Inclusion/Exclusion actions are applied to (e) an element or (f) a set. Inclusion and exclusion actions are applied to (h) an element or (g) a set.

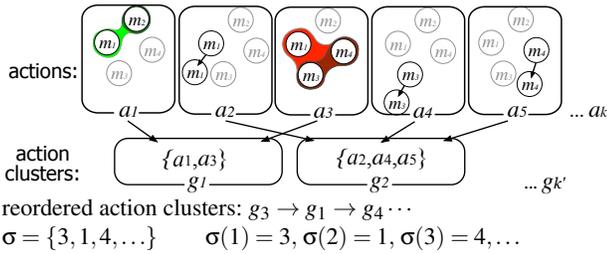


Figure 7: Reordering action clusters, each of which consists of individual actions.

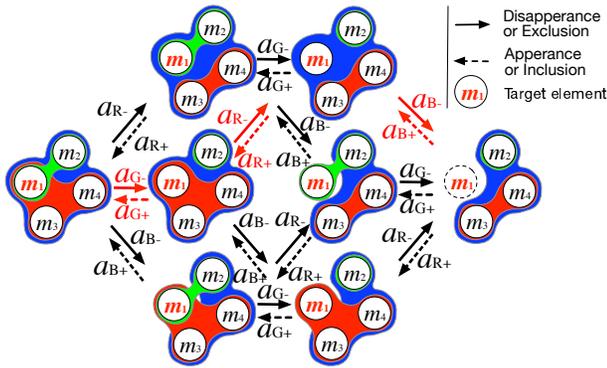


Figure 8: Optimizing the order of actions. From the leftmost diagram to the rightmost one, the sequence of actions a_{G-} , a_{R-} , and a_{B-} is the optimal order among possible action sequences.

- **Eye-tracking study:** We tracked the eye gaze behaviors of the subjects over synthesized animations to clarify the effects of action order optimization and grouping in our approach (Section 6.3).
- **Task-based analysis:** We recruited participants and asked them to perform a specific set of tasks through an online questionnaire,

to identify the advantages and limitations of our approach when compared with other relevant techniques (Section 6.4).

6.1. Data Generation

We prepared two real-world datasets for our experimental study:

- D.1 Map of flu spread across Europe:** We tried to visualize the spread of flu on the map in Europe during three weeks from November 27, 2017 to December 17, 2017 [flu]. More specifically, we visualized how the three different types of flu (i.e., Type A(H1), Type A(H3), and Type B) were prevalent in the respective European countries during that period of time. In this example, the elements of the set diagram correspond to countries on the map and are thus static.
- D.2 Authorships:** As a dynamic set diagram of relatively large size, we applied our method to visualize the trend of relationships between authors and research topics in the area of graph visualization. This data was obtained by partially extracting authors and keywords from the survey paper on graph visualization [VBW15]. To investigate the trend in this dataset, we identified four static set diagrams by sampling the temporal change in the authorship from 2008 to 2012, where the keywords and authors were extracted as elements and sets, respectively.

We employed these datasets for our eye-tracking experiment and online questionnaire as described later. Table 1 gives a summary of the datasets, including the numbers of elements, sets, actions, action clusters, and durations of the animations.

6.2. Computational Performance Study

Our prototype system was implemented on a desktop PC with Intel Xeon E5 CPU with 12 cores (2.7 GHz, 256 KB L2 Cache per core and 30 MB L3 Cache), 64GB RAM, and an AMD Dual Fire-Pro D700 GPU (3GB VRAM \times 2). The source code was written in C++ using the OpenGL, CGAL, IBM ILOG CPLEX, and GALib libraries for rendering, skeleton construction, integer programming, and genetic-based optimization, respectively. Table 2

Table 1: Numbers of elements ($\#\{E\}$), sets ($\#\{S\}$), actions ($\#\{A\}$), and action clusters ($\#\{AC\}$) and duration for each dataset.

Data	Figure	$\#\{E\}$	$\#\{S\}$	$\#\{A\}$	$\#\{AC\}$	Duration
D.1	Figure 12	12	3	22	16	15.0 sec
D.2	Figure 13	17	6	144	41	30.0 sec

Table 2: Performance.

Data	Rendering	Skeleton	Grouping	Ordering
D.1	15.63 fps	0.002 sec	0.058 sec	38.731 sec
D.2	3.82 fps	0.006 sec	0.228 sec	194.481 sec

shows the computational time of each experimental result. Each column represents skeleton graph construction (Section 4), action grouping (Section 5.2), optimal reordering of the action clusters (Section 5.3), and rendering (Section 4).

Figure 12 exhibits the temporal transition of flu spread across Europe on the map (D.1). We employed this example to evaluate how our approach can enhance the readability of the dynamic set diagram when the elements are fixed on the map domain. In contrast to simple linear interpolation, we can readily track individual changes in the set membership with our stepwise animation. Figure 13 shows the relationships between researchers and keywords in their publications (D.2). In this case, however, the keyword elements change their positions such that they come together to constitute a researcher set they belong to. The stepwise animation effectively facilitates us to identify the translations or exclusions/inclusions of keywords associated with the expertise of each researcher. (See the accompanying video also.)

6.3. Evaluation of Visualization Criteria on Gaze Shift

We conducted an eye-tracking experiment (with Tobii Pro X3-120 eye-tracker) to evaluate the effectiveness of the optimized order of actions and action clusters in terms of viewers' gaze behaviors. This lets us confirm that our approach only requires minimal gaze shift (C.3) and facilitate viewers to fully focus their attention on the changes in the animation.

Dataset. In this eye-tracking experiment, we employed the aforementioned two real-world datasets (D.1 and D.2) to help the participants intuitively interpret the context of the dynamic set diagrams. Moreover, we prepared these two datasets such that the elements are static and dynamic in the animated set diagrams, respectively.

Participants. We recruited 12 participants (three females and nine males; ages ranged from 19 to 24) with normal vision for this study. All of them were students majoring in computer science. Each participant was compensated with 2,000 JPY for his or her participation.

Tasks. The participants were requested to track the local change in the animations of the dynamic set diagrams with their eyes.

Conditions. We conducted the experiment with the four different types of animations (Table 3). Here, RO means that we just randomly arranged the temporal sequence of actions or action clusters in the animation, while OO indicates that we optimized the sequence using the proposed approach. Moreover, NG, RG, and OG

Table 3: Conditions of our eye-tracking experiment. The first and second two letters correspond to the ordering of actions or action clusters and method of grouping individual actions, respectively.

Condition	RO-NG	OO-NG	OO-RG	OO-OG
Ordering	Random	Optimized	Optimized	Optimized
Grouping	N/A	N/A	Random	Optimized

correspond to the cases in which actions are ordered without grouping, randomly grouped into clusters, and assembled into an optimal set of clusters with our formulation. In our experiment, we compare the results of conditions (RO-NG) and (OO-NG) for evaluating the effectiveness of ordering actions and (OO-RG) and (OO-OG) for that of grouping compatible actions.

Procedures. We asked the participants to visually track changes in the animations. During the eye-tracking experiment, we recorded the eye-gaze distributions and movements of the participants, as shown in Figure 9. Each participant observed eight animations (2 datasets \times 4 conditions) in total, while the order of animations was randomly changed as their frequencies were counterbalanced.

Results. Figure 9 shows the average heatmaps of the eye-gaze distributions for the keyframe images obtained using datasets D.1 and D.2. In the case of stepwise animations without the ordering optimization (RO-NG), the participants sometimes looked outside the animated region, and thus they sometimes failed to focus on local changes in the animations. In contrast, the stepwise animations with our ordering optimization (OO-NG) successfully prompted all participants to visually track the local animated regions. In comparison with the stepwise animations based on random action grouping (OO-RG), our optimized action grouping (OO-OG) facilitated the participants to concentrate on the local animated regions. This comparison suggests that our action grouping conditions successfully drew visual attention toward animated regions from the participants.

6.4. Evaluation of Readability on Analysis Tasks

In addition, we administered an online questionnaire with multiple-choice questions to investigate the readability of our visualization results when compared with conventional techniques.

Dataset. We again employed the aforementioned two real-world datasets; that is, the flu (D.1) and the authorship (D.2) datasets.

Participants. We recruited 38 participants (12 females and 26 males; ages ranged from 21 to 65) with normal vision for this study. Participants were primarily computer science majors and were compensated with 1,500 JPY for their participation.

Tasks. We selected four common tasks based on conventional taxonomy [AMA*14, BPF14] for our analysis:

- Q1 (*local*) Find elements that belong to a specific set.
- Q2 (*local*) Find sets that contain a specific element.
- Q3 (*global*) Find if a specific set has its elements changed.
- Q4 (*global*) Find sets having the most drastic changes in terms of the number of elements.

We prepared the first two as *local* tasks by referring to the element-related tasks supported by *KelpFusion* as summarized in the survey

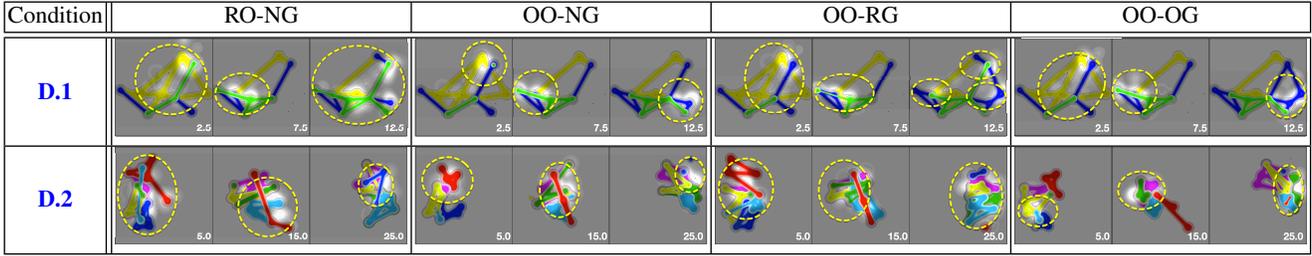


Figure 9: Average heatmaps of eye-gaze distribution. The white regions correspond to dense regions of the eye-gaze distribution, while the yellow circles indicate local animated regions. The white numbers on the lower right of the images represent the corresponding time (in seconds).

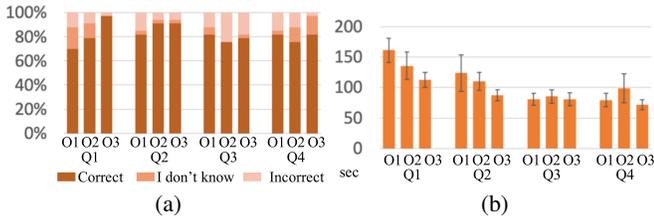


Figure 10: Results of the online questionnaire for **D.1**. (a) Answer categories and (b) task completion times for the four tasks. The error bars represent the standard errors.

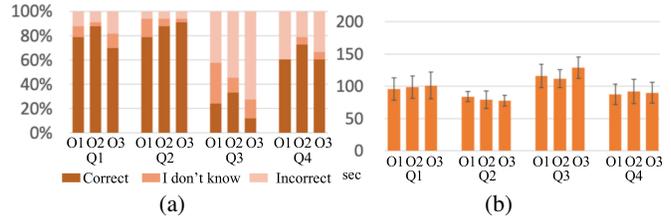


Figure 11: Results of the online questionnaire for **D.2**. (a) Answer categories and (b) task completion times for the four tasks. The error bars represent the standard errors.

on set diagram visualization [AMA*14], and the last two as *global* tasks of trend analysis employed in [BPF14]. Based on this, we thus designed four questions corresponding to each of the task for both datasets. (See the supplementary material for details.)

Conditions. We compared three types of animations. Three animations were prolonged equally for 15 seconds for **D.1** and 30 seconds for **D.2**.

- O1 *Simple linear interpolation (Parallel)*: All actions were animated linearly in parallel.
- O2 *3-step animation (3-steps)*: All actions were categorized into three types; that is, removal (A.1, A.3), translation (A.5), and addition (A.2, A.4) first, and the respective types of actions were animated individually in three steps, similarly to *Graph-Diaries* [BPF14].
- O3 *Stepwise animation (Ours)*: All actions were animated in a stepwise manner by applying the proposed approach.

Setup. During the questionnaire, participants could use the video progress bar to rewind or stop the animation. However, this will be measured in each task completion time for further analysis. We provided participants with the same animation content but with different color assignments for both datasets, and its reflections and rotations on **D.2** [BPF14]. We also counterbalanced the task order as well as the order of conditions for each task to avoid unwanted learning effects and influence from specific color assignments.

Procedures. Each participant was asked to answer two trial questions, including 24 formal questions (i.e., 2 datasets \times 3 conditions \times 4 tasks) in this questionnaire. We first described how to visually interpret the set structures and then train participants with two simple exercises. Once the participants understood the explanations

and moved on to the main questionnaire, they were asked to watch an animation video (stopping and rewinding allowed) and answered the corresponding multiple-choice questions. To avoid random selection, the participants could optionally select the “I don’t know” option. This study required approximately 40 minutes to complete.

Results. Figure 10 and Figure 11 show the statistical results for both datasets, each of which consists of answer categories (i.e., correct, wrong, and no answer) (Figure 10(a) and Figure 11(a)) and average task completion times (Figure 10(b) and Figure 11(b)). Note that, among the 38 participants, we eliminated four participants as outliers from the results due to their low accuracy ratios (less than 25 percent, which is close to random selection from the solution pools), and further screened them out if they spent more than 800 seconds on a task.

Our first example is the flu dataset. Recall that the positions of all elements here are fixed on the map as they correspond to the European countries. From the results presented in Figure 10, we found that *ours* achieved higher or equal accuracy compared to *parallel* and *3-steps* (Figure 10(a)), and participants could accomplish the tasks faster when using *ours* (Figure 10(b)). This tendency was especially strong for *local* tasks (Q1 and Q2), but less significant for *global* tasks (Q3 and Q4). *Parallel* in general leads to low accuracy and high completion times for *local* tasks, while *3-steps* is weak for *global* tasks. This is probably because important changes have been grouped together and additional time is thus needed to unravel the details.

Nonetheless, the advantage of stepwise animation changes when elements in the visualization begin to move. *Ours* still performs the best in terms of accuracy and completion time for Q2, while the accuracy decreases for Q1 (see Figure 11). For Q1, Q3, and Q4,

3-steps has better accuracy, while there is no significant advantage for the task completion time. During the analysis, we found that Q3 has extremely low accuracy for all approaches. This is because participants need to track individual changes in their membership of all elements during the spatial movement. We also consider that this is a difficult task requiring high mental workload.

6.5. Discussion

Based on the observations in Section 6.3 and Section 6.4, the eye-tracking experiment showed the effectiveness of gaze shift optimization in the sense that our stepwise animation successfully guided participants' visual attention to each step of the animation. We also learned from the task-based analysis that our stepwise animation provides better readability and help minimize the time to complete local analysis tasks (i.e., Q2)). This is because our approach decomposes complicated changes inherent in the dynamic set diagram to a sequence of actions, and thus the number of decomposed action clusters (i.e., the duration of the animated actions) effectively reflects the degree of complexity involved in the membership change (D.1). Meanwhile, *parallel* happened to be relatively competitive with *ours* for *global* tasks when the elements had fixed positions. We assume that participants can still identify important changes in the set coverage over time if the animation is slow enough to elucidate all the necessary actions.

On the contrary, in the analysis of the authorship dataset (D.2), the participants failed to conduct global tasks even with our stepwise animation. To further understand the data, we also conducted an ANOVA test to investigate the significance levels (p -values) on the answers of our questionnaire. Nonetheless, the difference between the three studied approaches are not significant enough ($p < 0.05$) to strongly support our hypothesis on the advantage of stepwise animation. This is probably due to the excessive overlap among the sets and too high speed of membership updates in the animation. Some participants claimed that the speed of the stepwise animation was too fast to follow, while suppressing conflicts among elements also helps to complete the tasks, which has been reflected in the task completion times. We expect to have a detailed study with a more sophisticated setting to clarify this question. Nevertheless, the between-subjects significance was smoothed in all conditions once having recruited a sufficient number of participants.

In summary, the stepwise animations are then considered more suitable for *local* tasks on the analysis of dynamic set diagrams, while more advanced analysis on understanding dynamic set diagrams should be further explored. As a limitation of our approach, it is still possible to misinterpret actions of translation as those of element exclusion/inclusion, which has been also suggested by the difference in the results between (D.1) and (D.2), in which the data elements were static and dynamic, respectively. Effectively controlling the spatial placement of elements and their associated sets while adjusting the animation speed still remains a technical challenge for future research.

7. Conclusion

In this paper, we have presented an approach to visualizing dynamic set diagrams by composing stepwise animations. The goal

of this study was to properly decompose atomic changes inherent in the dynamic set diagram for enhancing the visual readability of the individual temporal changes in the set membership. We achieved three appropriate design criteria for visualizing set diagrams by employing a contour-based representation. An algorithm is developed for optimizing the order of individual atomic changes while allowing the grouping of compatible changes in composing visually appealing stepwise animations. Our results together with an eye-tracking experiment demonstrate the effectiveness of the approach for visually tracking the detailed transitions in dynamic set diagrams, and a user study based on online questionnaires elucidates the advantages and possible issues of the proposed approach.

Acknowledgments

This work has been partially supported by JSPS KAKENHI (grant numbers 16H02825 and 15J00405) and EU Horizon 2020 Marie Skłodowska-Curie (grant number 747985). We also thank anonymous reviewers for their valuable comments and suggestions on our manuscript.

References

- [AMA*14] ALSALLAKH B., MICALLEF L., AIGNER W., HAUSER H., MIKSCH S., RODGERS P.: Visualizing Sets and Set-typed Data: State-of-the-Art and Future Challenges. In *Proceedings of the 16th Eurographics Conference on Visualization - State of The Art Reports (EuroVis '14)* (2014), pp. 1–21. doi:10.2312/eurovisstar.20141170. 2, 8, 9
- [AP13] ARCHAMBAULT D., PURCHASE H. C.: The "Map" in the Mental Map: Experimental Results in Dynamic Graph Drawing. *International Journal of Human-Computer Studies* 71, 11 (2013), 1044–1055. doi:10.1016/j.ijhcs.2013.08.004. 3
- [AP16] ARCHAMBAULT D., PURCHASE H. C.: Can animation support the visualisation of dynamic graphs? *Information Sciences* 330 (2016), 495–509. doi:10.1016/j.ins.2015.04.017. 3
- [APP11] ARCHAMBAULT D., PURCHASE H., PINAUD B.: Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics* 17, 4 (2011), 539–552. doi:10.1109/TVCG.2010.78. 3
- [ARRC11] ALPER B., RICHE N., RAMOS G., CZERWINSKI M.: Design Study of LineSets, a Novel Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2259–2267. doi:10.1109/TVCG.2011.186. 2
- [BBDW14] BECK F., BURCH M., DIEHL S., WEISKOPF D.: The State of the Art in Visualizing Dynamic Graphs. In *Proceedings of the 16th Eurographics Conference on Visualization - State of The Art Reports (EuroVis '14)* (2014), pp. 83–103. doi:10.2312/eurovisstar.20141174. 3
- [BHS12] BAUHOFF V., HUFF M., SCHWAN S.: Distance matters: Spatial contiguity effects as trade-off between gaze switches and memory load. *Applied Cognitive Psychology* 26, 6 (2012), 863–871. doi:10.1002/acp.2887. 2, 4
- [BIM12] BRANDES U., INDEKOEFER N., MADER M.: Visualization Methods for Longitudinal Social Networks and Stochastic Actor-Oriented Modeling. *Social Networks* 34, 3 (2012), 291–308. doi:10.1016/j.socnet.2011.06.002. 3
- [BLA*11] BREMM S., LANDESBERGER T. V., ANDRIENKO G., ANDRIENKO N., SCHRECK T.: Interactive Analysis of Object Group Changes over Time. In *Proceedings of the International Workshop on Visual Analytics (EuroVA '11)* (2011), pp. 41–44. doi:10.2312/PE/EuroVAST/EuroVA11/041-044. 2

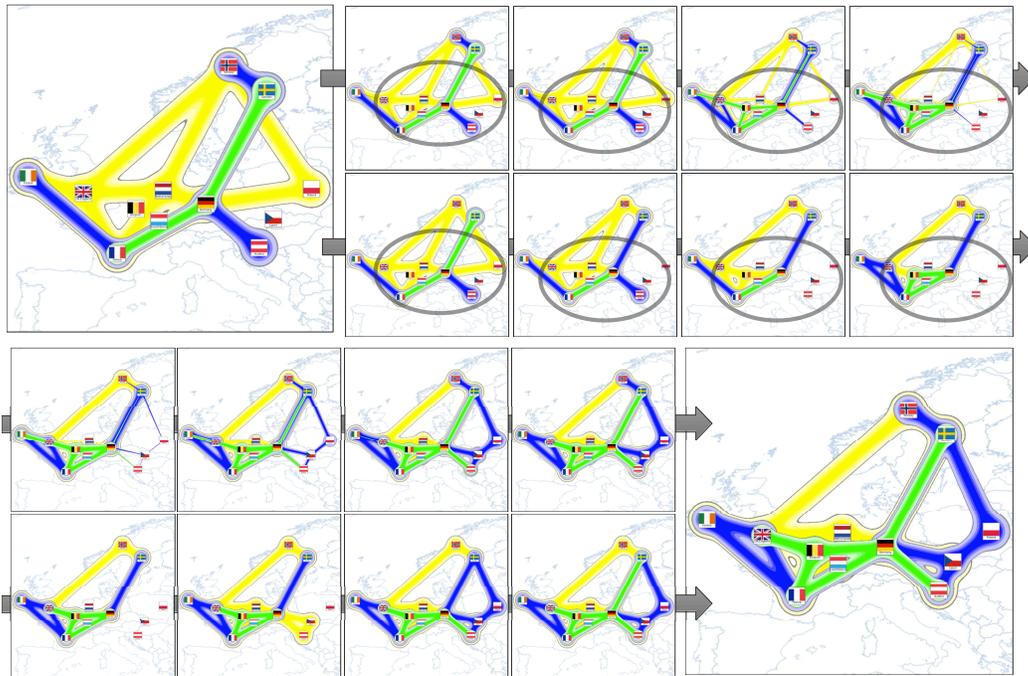


Figure 12: Snapshots of the stepwise animation of the flu dataset *D.1*. This animation sequence contains 22 actions and 16 action clusters.

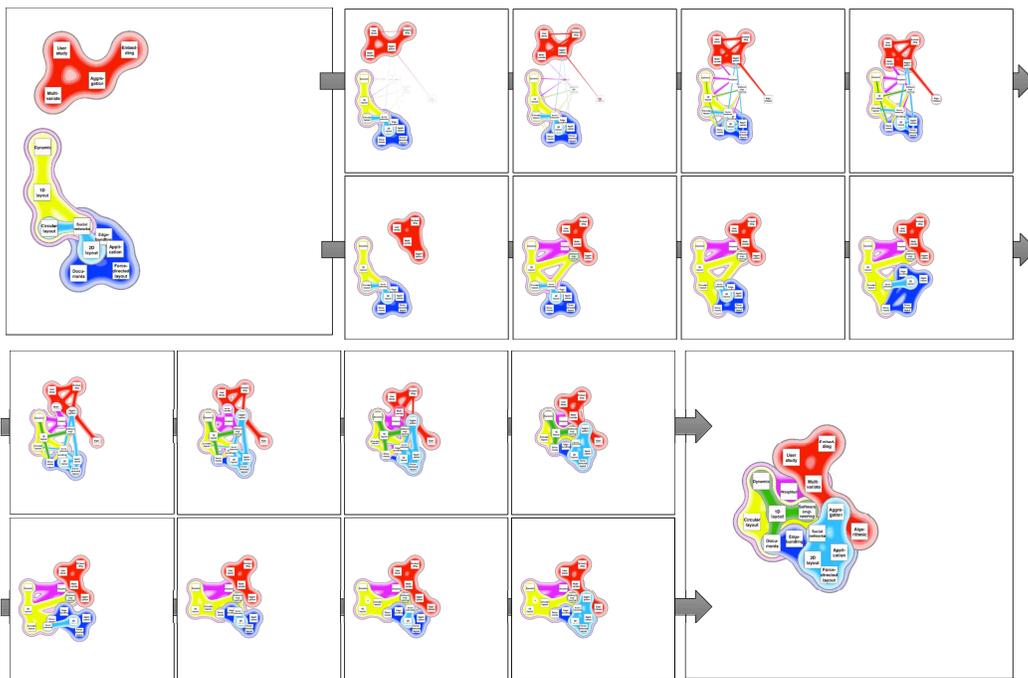


Figure 13: Snapshots of the stepwise animation of the authorship dataset *D.2*. The sequence contains 48 actions and 13 action clusters.

[BPF14] BACH B., PIETRIGA E., FEKETE J.-D.: GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics* 20, 5 (2014), 740–754. doi:10.1109/TVCG.2013.254. 3, 4, 6, 8, 9

[BT09] BYELAS H., TELEA A.: Visualizing Metrics on Areas of In-

terest in Software Architecture Diagrams. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '09)* (2009), pp. 33–40. doi:10.1109/PACIFICVIS.2009.4906835. 2

[CPC09] COLLINS C., PENN G., CARPENDALE S.: Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations.

- IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1009–1016. doi:10.1109/TVCG.2009.122. 2
- [dBMS11] DE BERG M., MEULEMANS W., SPECKMANN B.: Delineating Imprecise Regions via Shortest-Path Graphs. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '11)* (2011), pp. 271–280. doi:10.1145/2093973.2094010. 4
- [DEKB*14] DINKLA K., EL-KEBIR M., BUCUR C.-I., SIDERIUS M., SMIT M. J., WESTENBERG M., KLAU G. W.: eXamine: Exploring Annotated Modules in Networks. *BMC Bioinformatics* 15, 1 (2014), 201. doi:10.1186/1471-2105-15-201. 2
- [DvKSW12] DINKLA K., VAN KREVELD M. J., SPECKMANN B., WESTENBERG M. A.: Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum* 31, 3pt1 (2012), 875–884. doi:10.1111/j.1467-8659.2012.03080.x. 2
- [FA04] FRISHMAN Y., AYELLET T.: Dynamic Drawing of Clustered Graphs. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)* (2004), pp. 191–198. doi:10.1109/INFVIS.2004.18. 3
- [FE01] FRIEDRICH C., EADES P.: The Marey Graph Animation Tool Demo. In *Proceedings of the 8th International Symposium on Graph Drawing (GD '01)* (2001), Springer Lecture Notes in Computer Science, pp. 396–406. doi:10.1007/3-540-44541-2_37. 3
- [FH02] FRIEDRICH C., HOULE M.: Graph Drawing in Motion II. *Journal of Graph Algorithms and Applications* 6, 3 (2002), 220–231. doi:10.1007/3-540-45848-4_18. 3
- [flu] Flu news europe. <https://flunewseurope.org/>. 7
- [HMdS*15] HEBERLE H., MEIRELLES G. V., DA SILVA F. R., TELLES G. P., MINGHIM R.: InteractiVenn: a Web-Based Tool for the Analysis of Sets Through Venn diagrams. *BMC bioinformatics* 16, 1 (2015), 169. doi:10.1186/s12859-015-0611-3. 2
- [LSCL10] LIN Y.-R., SUN J., CAO N., LIU S.: ContextTour: Contextual Contour Visual Analysis on Dynamic Multi-Relational Clustering. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM '10)* (2010), pp. 418–429. doi:10.1137/1.9781611972801.37. 3
- [MKY12] MASHIMA D., KOBouROV S., YIFAN H.: Visualizing Dynamic Data with Maps. *IEEE Transactions on Visualization and Computer Graphics* 18, 9 (2012), 1424–1437. doi:10.1109/TVCG.2011.288. 3
- [MRS*13] MEULEMANS W., RICHE N. H., SPECKMANN B., ALPER B., DWYER T.: KelpFusion: A Hybrid Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics* 19, 11 (2013), 1846–1858. doi:10.1109/TVCG.2013.76. 2, 4
- [Nor96] NORTH S. C.: Incremental layout in DynaDAG. In *Proceedings of the 3rd International Symposium on Graph Drawing (GD '96)* (1996), Springer Lecture Notes in Computer Science, pp. 409–418. doi:10.1007/BFb0021824. 3
- [RD10] RICHE N. H., DWYER T.: Untangling Euler Diagrams. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1090–1099. doi:10.1109/TVCG.2010.210. 2
- [RM13] RUFIANGE S., MCGUFFIN M. J.: DiffAni: Visualizing Dynamic Graphs with a Hybrid of Difference Maps and Animation. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2556–2565. doi:10.1109/TVCG.2013.149. 3
- [RMF04] RODGERS P., MUTTON P., FLOWER J.: Dynamic Euler Diagram Drawing. In *Proceedings of the IEEE Symposium on Visual Languages - Human Centric Computing* (2004), pp. 147–156. doi:10.1109/VLHCC.2004.21. 2
- [Rod14] RODGERS P.: A survey of Euler diagrams. *Journal of Visual Languages & Computing* 25, 3 (2014), 134–155. doi:10.1016/j.jvlc.2013.08.006. 2
- [SA08] SIMONETTO P., AUBER D.: Visualise Undrawable Euler Diagrams. In *Proceedings of the 12th International Conference Information Visualisation (IV '08)* (2008), pp. 594–599. doi:10.1109/IV.2008.78. 2
- [VBW15] VEHLow C., BECK F., WEISKOPF D.: The State of the Art in Visualizing Group Structures in Graphs. In *Proceedings of the 17th Eurographics Conference on Visualization - State of The Art Reports (EuroVis '15)* (2015), pp. 21–40. doi:10.2312/eurovisstar.20151110. 7

Optimizing Stepwise Animation in Dynamic Set Diagrams (Supplementary Material)

Kazuyo Mizuno¹, Hsiang-Yun Wu², Shigeo Takahashi³, and Takeo Igarashi⁴

¹Yahoo Japan Corporation, Japan

²TU Wien, Austria

³The University of Aizu, Japan

⁴The University of Tokyo, Japan

Appendix A: Formulation of Action Grouping

To find an optimal collection of subsets of actions, we solve a variant of the *set cover problem* that finds an optimal collection of subsets that covers all elements while minimizing a predefined cost function. In the proposed approach, we duplicate all actions except for translation type actions as two different groups; that is, actions $\{a_1^p, a_2^p, \dots, a_k^p\}$ that are performed prior to the translation of the corresponding elements, and actions $\{a_1^q, a_2^q, \dots, a_k^q\}$ that are performed after the translation. Recall that k is the number of actions. We then exhaustively enumerate a collection of all possible subsets of compatible actions in such a way that every pair of actions in each subset satisfies the conditions. Our set cover problem can be solved as an integer programming problem by minimizing:

$$\sum_{j=1}^r w_j u_j, \quad (\text{S.1})$$

while satisfying:

$$\sum_{j=1}^r (b_{ij}^p + b_{ij}^q) u_j = 1 \quad \forall i, \quad (\text{S.2})$$

where r indicates the total number of such possible subsets of compatible actions. Here, the binary value b_{ij}^p or b_{ij}^q is 1 only if a_i^p or a_i^q , respectively, is contained in the j -th subset. Note that the duplicated actions a_i^p and a_i^q are incompatible with each other, and thus only one of the two actions is activated in the entire animation. Furthermore, w_j is a weight value for the j -th subset and is computed as the weighted sum of the average distance between all pairs of elements in that subset and the average distance between all pairs of sets along the depth axis in that subset. In this formulation, u_j is 1 if the corresponding j -th subset is selected as a cluster of actions; thus, by referring to the solution, we can group actions into several clusters to reduce the total number of animation steps.

Note that in our experimental results, the maximum number of actions that coexist in a single animation step is five, following the evaluation of dynamic graph animation conducted by Archambault et al. [API13].

Appendix B: Optimization of the Action-Cluster Order

Weights of Cost Function

Note that α , β , γ , and δ are the weight values of the cost functions for ordering action clusters (in Section 5.3), respectively, and, unless stated otherwise, are set as $\alpha = \beta = \gamma = 1.0$ and $\delta = 10000$. For adjusting the weight values, our guideline suggests that we should increase α if the dynamic set diagram has a large amount of translation for its elements, β if the diagram has a large number of exclusions and inclusions, and γ if it intrinsically contains frequent change in the depth order of sets. The last weight δ is set to be relatively large since the corresponding function F_4 penalizes the unnecessary splitting of sets in the optimization.

Computational Cost

Our optimization seeks the best permutation of action clusters that optimizes the total cost function and thus can be formulated as a variant of the traveling salesman problem. With respect to algorithm complexity, the worst computational complexity of the cost functions at one permutation is $\mathcal{O}(k^2 + nmk')$, where k and k' are the number of primitive actions and action clusters, respectively, and n and m are the number of elements and sets, respectively. This means that the computational complexity of all possible permutations in the proposed method is $\mathcal{O}(k'!(k^2 + nmk'))$. We employed a simple *genetic algorithm* (GA) [Gol89] to find the best possible solution within a reasonable period of time. This was accomplished by encoding the sequence of action cluster IDs as a value-encoding chromosome and performing genetic-based optimization by evolving the collection of elite chromosomes.

Appendix C: Implementation Details of Rendering Set Diagrams

Contour Generation

To draw a smooth boundary for each set, we extract the *contours* of the 2D scalar field as *implicit curves* [WWI07] to properly associate spatial regions with the constructed skeleton graph.

Given a set of data samples m_1, m_2, \dots, m_n , a blending function

at a point m is commonly defined as follows:

$$f_h(m) = \frac{1}{n} \sum_{i=1}^n K_h(m - m_i), \quad (\text{S.3})$$

where K is a kernel function and h is its bandwidth parameter. To expand the skeleton graph to a contour region, we first calculate the entire scalar field as the sum of individual kernel functions associated with its member elements. Then, we assign a line kernel function $L(e)$ to each edge e in the skeleton graph and incorporate the expanding effect into the scalar field to reinforce the scalar field within the set region. Note that we employ the previously proposed line kernel function [DH11] as follows:

$$L_h(e) = \int_0^1 K_h(m - ((1 - \phi)m_s + \phi m_e)) d\phi, \quad (\text{S.4})$$

where m_s and m_e are the start and end positions of an edge. Specifically, we use a Gaussian window function as the kernel function. After accumulating all kernel functions, we can compute the entire scalar field by following the work of Daae Lampe and Hauser [DH11]. Finally, we extract a contour line for each set by sectioning the scalar field at a specific value.

In the proposed approach, we place a set that is smaller in terms of its size over larger sets to avoid unwanted occlusions. For this purpose, we increase the bandwidth parameter by a constant value as the depth of the corresponding set becomes more.

Color Assignment of Sets

In the proposed system, we assign a unique color to each set and apply gradation according to the corresponding scalar field value of the set. This is implemented by first selecting the hue value of the HSV color space according to the ID of the corresponding set and then changing the saturation value according to the scalar field value within that set. Thus, the HSV color (i.e., (hue, saturation, value)) at pixel p within the set s_i can be calculated as follows:

$$C(s_i, p) = \left(\frac{2\pi i}{l}, f(p), 1.0 \right), \quad (\text{S.5})$$

where l is the total number of sets, i is the ID of the current set, and $f(p)$ represents the normalized scalar field value at pixel p ($0 \leq f(p) \leq 1$). To fully discriminate between different colors of adjacent sets, we set the color of the contours to gray.

Interpolation for Animated Transition

To synthesize the stepwise animation of the dynamic set diagrams, we must define how to smoothly interpolate individual nodes and edges in the skeleton graph according to their appearance/disappearance. Rather than morphing between contour lines of sets directly, we gradually change the bandwidth values of the kernel functions associated with the nodes and length of the edges according to the action types. In practice, we change three visual characteristics of the set diagrams before and after action a_i ; that is, the contour lines associated with target element $m_{M(i)}$, edges incident to $m_{M(i)}$, and the depth ordering of set $s_{S(i)}$. Recall that $M(i)$ and $S(i)$ are the IDs of the elements and sets associated with the i -th action a_i , respectively. Note that in all cases, we introduce ease

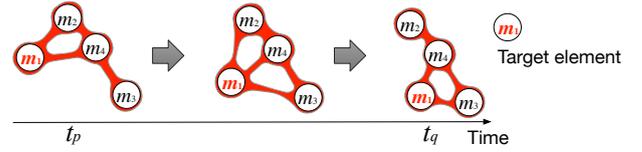


Figure S.1: Example of a union graph.

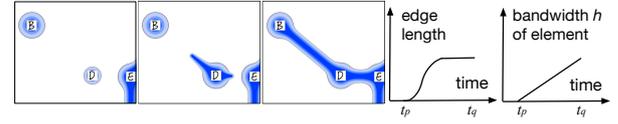


Figure S.2: Animation steps associated with edge insertion.

in/out effects [Las87] in the animation to augment the perceptual smoothness of the dynamic set diagrams.

When interpolating the contour lines around the target element, we linearly change the bandwidth value of the associated kernel functions (Eq. (S.3) and Eq. (S.4)) during the transition of a_i . Note that we set the bandwidth value h to 0 if the target element does not exist before or after the action.

When inserting/removing edges incident to the target element, we employ a union graph of the two skeleton graphs at t_p and t_q using the method developed by Diehl et al. [DGK01, DG02] to maximally preserve the mental map. In the middle of the animation step of the target action, we render the edges incident to the target element within the union graph, as shown in Figure S.1. We also smoothly control the length of the edges in proportion to the change in bandwidth, as shown in Figure S.2.

As described previously, in the proposed approach, the depths of all the sets are sorted according to size. In the actual implementation, each set implicitly has a real variable as its depth value, which allows us change the depth sequence of the sets by linearly interpolating the depth values, as shown in Figure S.3. When two sets have sufficiently close depth values, we also smoothly change the set diagram color by applying alpha blending in the overlapped region (Figure S.3).

Appendix D: Questionnaires for User Study

In the user study, the questions of each dataset were set as follows.

Flu

- Q1) Find the countries that were infected with flu of Type A(H3) and Type B.
- Q2) Find the types of viruses that infected the countries of Poland, the Czech Republic, and Austria (at least once).
- Q3) Find countries that were always covered by the same types of virus.
- Q4) Find the country that has the strongest impact on distributing virus to the other countries.

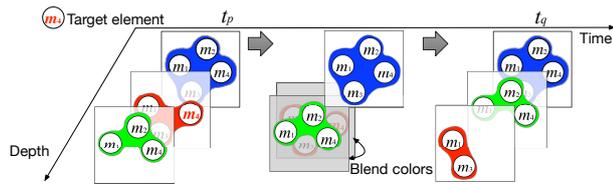


Figure S.3: Smoothly changing the depth order and the color of sets.

Authorship

- Q1) Find the research topic that was investigated by the most researchers.
- Q2) Find the researcher whose research topics were completely contained by other researchers.
- Q3) Find a researcher who always concentrated on a specific set of research topics.
- Q4) Find the researcher who increased her/his research topics the most.

References

- [AP13] ARCHAMBAULT D., PURCHASE H. C.: The "Map" in the Mental Map: Experimental Results in Dynamic Graph Drawing. *International Journal of Human-Computer Studies* 71, 11 (2013), 1044–1055. doi:10.1016/j.ijhcs.2013.08.004. 1
- [DG02] DIEHL S., GÖRG C.: Graphs, They are Changing Dyamic Graph Drawing for a Sequence of Graphs. In *Proceedings of the 9th International Symposium on Graph Drawing (GD '02)* (2002), vol. 2528, Springer Lecture Notes in Computer Science, pp. 23–31. doi:10.1007/3-540-36151-0_3. 2
- [DGK01] DIEHL S., GÖRG C., KERREN A.: Preserving the Mental Map Using Foresighted Layout. In *Proceedings of the 3rd Joint Eurographics/IEEE VGTC Conference on Visualization (EuroVis '01)* (2001), vol. 1, pp. 175–184. doi:10.1007/978-3-7091-6215-6_19. 2
- [DH11] DAAE LAMPE O., HAUSER H.: Interactive Visualization of Streaming Data with Kernel Density Estimation. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '11)* (2011), pp. 171–178. doi:10.1109/PACIFICVIS.2011.5742387. 2
- [Gol89] GOLDBERG D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Professional, 1989. 1
- [Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), SIGGRAPH '87, pp. 35–44. doi:10.1145/37401.37407. 2
- [WWI07] WATANABE N., WASHIDA M., IGARASHI T.: Bubble clusters: An Interface for Manipulating Spatial Aggregation of Graphical Objects. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)* (2007), pp. 173–182. doi:10.1145/1294211.1294241. 1